Temporal Network Optimization Subject to Connectivity Constraints*

George B. Mertzios† Othon Michail‡ Paul G. Spirakis§

Abstract

In this work we consider *temporal networks*, i.e. networks defined by a *labeling* λ assigning to each edge of an *underlying graph* G a set of *discrete* time-labels. The labels of an edge, which are natural numbers, indicate the discrete time moments at which the edge is available. We focus on *path problems* of temporal networks. In particular, we consider *time-respecting* paths, i.e. paths whose edges are assigned by λ a strictly increasing sequence of labels. We begin by giving two efficient algorithms for computing shortest time-respecting paths on a temporal network. We then prove that there is a *natural analogue of Menger's theorem* holding for arbitrary temporal networks. Finally, we propose two *cost minimization parameters* for temporal network design. One is the *temporality* of G, in which the goal is to minimize the maximum number of labels of an edge, and the other is the *temporal cost* of G, in which the goal is to minimize the total number of labels used. Optimization of these parameters is performed subject to some *connectivity constraint*. We prove several lower and upper bounds for the temporality and the temporal cost of some very basic graph families such as rings, directed acyclic graphs, and trees.

Keywords: Temporal network, graph labeling, Menger's theorem, optimization, temporal connectivity, hardness of approximation.

1 Introduction

A *temporal* (or *dynamic*) *network* is, loosely speaking, a network that changes with time. This notion encloses a great variety of both modern and traditional networks such as information and communication networks, social networks, transportation networks, and several physical systems. In the literature of traditional communication networks, the network topology is rather static, i.e. topology modifications are rare and they are mainly due to link failures and congestion. However, most modern communication networks such as mobile ad hoc, sensor, peer-to-peer, opportunistic, and delay-tolerant networks are inherently dynamic and it is often the case that this dynamicity is of a very high rate. In social networks, the topology usually represents the social connections between a group of individuals and it changes as the social relationships between the individuals are updated, or as existing individuals leave, or new individuals enter the group. In a transportation network, there is usually some fixed network of routes and a set of transportation units moving over these routes and dynamicity refers to the change of the positions of the transportation units in the network as time passes. Physical systems of interest may include several systems of interacting particles.

In this work, embarking from the foundational work of Kempe *et al.* [15], we consider *discrete time*, that is, we consider networks in which changes occur at discrete moments in time, e.g. days. This choice is not only a very natural abstraction of many real systems but also gives to the

*This work was supported in part by (i) the project "Foundations of Dynamic Distributed Computing Systems" (FOCUS) which is implemented under the "ARISTEIA" Action of the Operational Programme "Education and Lifelong Learning" and is co-funded by the European Union (European Social Fund) and Greek National Resources, (ii) the FET EU IP project MULTIPLEX under contract no 317532, and (iii) the EPSRC Grants EP/P020372/1, EP/P02002X/1, and EP/K022660/1. A preliminary version of this work has appeared in ICALP 2013 [18].

†Department of Computer Science, Durham University, UK. Email: george.mertzios@durham.ac.uk

‡Department of Computer Science, University of Liverpool, UK. Email: Othon.Michail@liverpool.ac.uk

§Department of Computer Science, University of Liverpool, UK and Computer Technology Institute & Press "Diophantus" (CTI). Email: P.Spirakis@liverpool.ac.uk

resulting models a purely combinatorial flavor. In particular, we consider those networks that can be described via an underlying graph G and a labeling λ assigning to each edge of G a (possibly empty) set of discrete labels. Note that this is a generalization of the single-label-per-edge model used in [15], as we allow many time-labels to appear on an edge. These labels are drawn from the natural numbers and indicate the discrete moments in time at which the corresponding connection is available. For example, in the case of a communication network, availability of a communication link at some time t may mean that a communication protocol is allowed to transmit a data packet over that link at time t.

In this work, we initiate the study of the following fundamental network design problem: "*Given an underlying (di)graph G, assign labels to the edges of G so that the resulting temporal graph $\lambda(G)$ minimizes some parameter while satisfying some connectivity property*". In particular, we consider two cost optimization parameters for a given graph G. The first one, called *temporality* of G, measures the maximum number of labels that an edge of G has been assigned. The second one, called *temporal cost* of G, measures the total number of labels that have been assigned to all edges of G (i.e. if $|\lambda(e)|$ denotes the number of labels assigned to edge e, we are interested in $\sum_{e\in E}|\lambda(e)|$). That is, if we interpret the number of assigned labels as a measure of *cost*, the temporality (resp. the temporal cost) of G is a measure of the decentralized (resp. centralized) cost of the network, where only the cost of individual edges (resp. the total cost over all edges) is considered. Each of these two cost measures can be minimized subject to some particular connectivity property $\mathcal{P}$ that the temporal graph $\lambda(G)$ has to satisfy. In this work, we consider two very basic connectivity properties. The first one, that we call the *all paths* property, requires the temporal graph to preserve every simple path of its underlying graph, where by "preserve a path of G" we mean in this work that the labeling should provide at least one strictly increasing sequence of labels on the edges of that path, in which case we also say that the path is *time-respecting*.

Before describing our second connectivity property let us give a simple illustration of temporality minimization. We are given a directed ring $u_1, u_2, \ldots, u_n$ and we want to determine the temporality of the ring subject to the all paths property. That is, we want to find a labeling λ that preserves every simple path of the ring and at the same time minimizes the maximum number of labels of an edge. Looking at Figure 1, it is immediate to observe that an increasing sequence of labels on the edges of path P_1 implies a decreasing pair of labels on edges (u_{n-1}, u_n) and (u_1, u_2). On the other hand, path P_2 uses first (u_{n-1}, u_n) and then (u_1, u_2) thus it requires an increasing pair of labels on these edges. It follows that in order to preserve both P_1 and P_2 we have to use a second label on at least one of these two edges, thus the temporality is at least 2. Next, consider the labeling that assigns to each edge (u_i, u_{i+1}) the labels $\{i, n+i\}$, where $1 \leq i \leq n$ and $u_{n+1} = u_1$. It is not hard to see that this labeling preserves all simple paths of the ring. Since the maximum number of labels that it assigns to an edge is 2, we conclude that the temporality is also at most 2. In summary, the temporality of preserving all simple paths of a directed ring is 2.

The other connectivity property that we define, called the *reach* property, requires the temporal graph to preserve a path from node u to node v whenever v is reachable from u in the underlying graph. Furthermore, the minimization of each of our two cost measures can be affected by some problem-specific constraints on the labels that we are allowed to use. We consider here one of the most natural constraints, namely an upper bound of the *age* of the constructed labeling λ, where the age of a labeling λ is defined to be equal to the maximum label of λ minus its minimum label plus 1. Now the goal is to minimize the cost parameter, e.g. the temporality, satisfy the connectivity property, e.g. *all paths*, and additionally guarantee that the age does not exceed some given natural k. Returning to the ring example, it is not hard to see, that if we additionally restrict the age to be at most $n-1$ then we can no longer preserve all paths of a ring using at most 2 labels per edge. In fact, we must now necessarily use the worst possible number of labels, i.e. $n-1$ on every edge.

Minimizing such parameters may be crucial as, in most real networks, making a connection available and maintaining its availability does not come for free. For example, in wireless sensor networks the cost of making edges available is directly related to the power consumption of keeping nodes awake, of broadcasting, of listening to the wireless channel, and of resolving the resulting

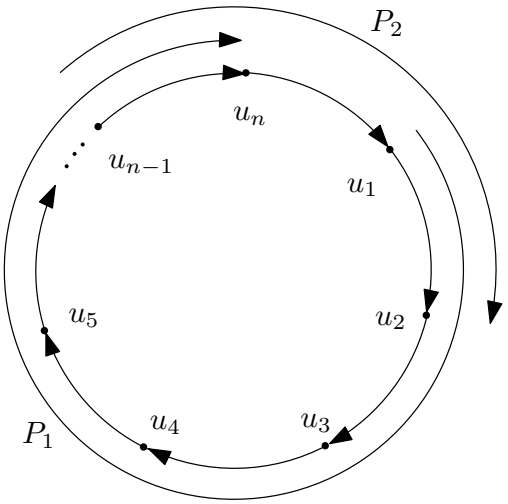

Figure 1: Path P_2 forces a second label to appear on either (u_{n-1}, u_n) or (u_1, u_2).

communication collisions. The same holds for transportation networks where the goal is to achieve good connectivity properties with as few transportation units as possible. At the same time, such a study is important from a purely graph-theoretic perspective as it gives some first insight into the structure of specific families of temporal graphs. To make this clear, consider again the ring example. Proving that the temporality of preserving all paths of a ring is 2 at the same time proves the following. If a *temporal ring* is defined as a ring in which all nodes can communicate clockwise to all other nodes via time-respecting paths then *no temporal ring exists with fewer than* $n + 1$ *labels*. This, though an easy one, is a structural result for temporal graphs. Finally, we believe that our results are a first step towards answering the following fundamental question: "*To what extent can algorithmic and structural results of graph theory be carried over to temporal graphs?*". For example, is there an analogue of Menger's theorem for temporal graphs? One of the results of the present work is an affirmative answer to the latter question.

1.1 Related Work

Labeled Graphs. Labeled graphs have been widely used in Computer Science and Mathematics, e.g. in Graph Coloring [23]. In our work, labels correspond to moments in time and the properties of labeled graphs that we consider are naturally *temporal properties*. Note, however, that any property of a graph labeled from a discrete set of labels corresponds to some temporal property if interpreted appropriately. For example, a proper edge-coloring, i.e. a coloring of the edges in which no two adjacent edges share a common color, corresponds to a temporal graph in which no two adjacent edges share a common label, i.e. no two adjacent edges ever appear at the same time. Though we focus on properties with natural temporal meaning, our definitions are generic and do not exclude other, yet to be defined, properties that may prove important in future applications.

Single-label Temporal Graphs and Menger's Theorem. The model of temporal graphs that we consider in this work is a direct extension of the single-label model studied in [4] and [15] to allow for many labels per edge. The main result of [4] was that in single-label networks the max-flow min-cut theorem holds with unit capacities for time-respecting paths. In [15], Kempe *et al.*, among other things, proved that a fundamental property of classical graphs does not carry over to their temporal counterparts. In particular, they proved that there is no analogue of Menger's theorem, at least in its original formulation, for arbitrary single-label temporal networks and that the computation of the number of node-disjoint s-t time-respecting paths is NP-complete. *Menger's theorem* states that the maximum number of node-disjoint s-t paths is equal to the minimum number of nodes needed to separate s from t (see [5]). In this work, we go a step ahead showing that if one reformulates Menger's theorem in a way that takes time into account then a very natural temporal analogue of Menger's theorem is obtained. Both of the above papers, consider a path as *time-respecting* if

its edges have non-decreasing labels. In the present work, we depart from this assumption and consider a path as time-respecting if its edges have *strictly increasing* labels. Our choice is very well motivated by recent work in dynamic communication networks. If it takes one time unit to transmit a data packet over a link then a packet can only be transmitted over paths with strictly increasing availability times.

Continuous Availabilities (Intervals). Some authors have assumed that an edge may be available for a whole time-interval $[t_1, t_2]$ or several such intervals and not just for discrete moments as we assume here. This is a clearly natural assumption but the techniques used in those works are quite different from those needed in the discrete case [11, 28].

Dynamic Distributed Networks. In recent years, there is a growing interest in distributed computing systems that are inherently dynamic. This has been mainly driven by the advent of low-cost wireless communication devices and the development of efficient wireless communication protocols. Apart from the huge amount of work that has been devoted to applications, there is also a steadily growing concrete set of foundational work. A notable set of works has studied (distributed) computation in *worst-case* dynamic networks in which the topology may change arbitrarily from round to round subject to some constraints that allow for bounded end-to-end communication [10, 17, 22, 25]. Population protocols [2] and variants [20] are collections of finite-state agents that move arbitrarily like a soup of particles and interact in pairs when they come close to each other. The goal is there for the population to compute (i.e. agree on) something useful in the limit in such an adversarial setting. Another interesting direction assumes that the dynamicity of the network is a result of randomness. Here the interest is on determining "good" properties of the dynamic network that hold with high probability, such as small (temporal) diameter, and on designing protocols for distributed tasks [3, 7]. For introductory texts on the above lines of research in dynamic distributed networks the reader is referred to [6, 21, 26].

Distance Labeling. A distance labeling of a graph G is an assignment of unique labels to the vertices of G so that the distance between any two vertices can be inferred from their labels alone. The goal is to minimize some parameter of the labeling and to provide a (hopefully fast) decoder algorithm for extracting a distance from two labels [13, 14]. There are several differences between a distance labeling and the time-labelings that we consider in this work. First of all, a distance labeling is being assigned on the vertices and not on the edges. Moreover, in distance labeling, one usually seeks the most compact set of labels (in binary length) that still guarantees efficient decoding. That is, the labeling parameter to be minimized is the binary length of an appropriate encoding, which is quite different from our cost parameters. Finally, the optimization constraint there is efficient decoding while in our case the constraints have to do with connectivity properties of the labeled graph.

Also, we encourage the interested reader to see [19] for a recent introductory text on the recent algorithmic progress on temporal graphs.

1.2 Contribution

In §2, we formally define the model of temporal graphs under consideration and provide all further necessary definitions. The rest of the paper is partitioned into two parts. Part I focuses on journey problems for temporal graphs. In particular, in §3, we give two efficient algorithms for computing shortest time-respecting paths. Then in §4 we present an analogue of Menger's theorem which we prove valid for arbitrary temporal graphs. We apply our Menger's analogue to simplify the proof of a recent result on distributed token gathering. Part II studies the problem of designing a temporal graph optimizing some parameters while satisfying some connectivity constraints. Specifically, in §5 we formally define the temporality and temporal cost optimization metrics for temporal graphs. In §5.1, we provide several upper and lower bounds for the temporality of some fundamental graph families such as rings, directed acyclic graphs (DAGs), and trees, as well as an interesting trade-off between the temporality and the age of rings. Furthermore, we provide in §5.2 a generic method for computing a lower bound of the temporality of an arbitrary graph w.r.t. the *all paths* property, and we illustrate its usefulness in cliques, close-to-complete bipartite subgraphs, and planar graphs.

In §5.3, we consider the temporal cost of a digraph G w.r.t. the *reach* property, when additionally the age of the resulting labeling $\lambda(G)$ is restricted to be the smallest possible. We prove that this problem is hard to approximate, i.e. there exists no PTAS unless P=NP. To prove our claim, we first prove (which may be of interest in its own right) that the Max-XOR(3) problem is APX-hard via a PTAS reduction from Max-XOR. In the Max-XOR(3) problem, we are given a 2-CNF formula ϕ, every literal of which appears in at most 3 clauses, and we want to compute the greatest number of clauses of ϕ that can be simultaneously XOR-satisfied. Then we provide a PTAS reduction from Max-XOR(3) to our temporal cost minimization problem. On the positive side, we provide an $(r(G)/n)$-factor approximation algorithm for the latter problem, where $r(G)$ denotes the total number of reachabilities in G. Finally, in §6 we conclude and give further research directions that are opened by our work.

2 Preliminaries

2.1 A Model of Temporal Graphs

Given a (di)graph $G = (V, E)$, [1] a *labeling* of G is a mapping $\lambda : E \to 2^{\mathbb{N}}$, that is, a labeling assigns to each edge of G a (possibly empty) [2] set of natural numbers, called *labels*.

Definition 1 *Let $G = (V, E)$ be a (di)graph and λ be a labeling of G. Then $\lambda(G)$ is the* temporal graph *(or* dynamic graph [3]*) of G with respect to λ. Furthermore, G is the* underlying graph *of $\lambda(G)$.*

We denote by $\lambda(E)$ the multiset of all labels assigned to the underlying graph by the labeling λ and by $|\lambda| = |\lambda(E)|$ their cardinality (i.e. $|\lambda| = \sum_{e \in E} |\lambda(e)|$). We also denote by $\lambda_{\min} = \min\{l \in \lambda(E)\}$ the minimum label and by $\lambda_{\max} = \max\{l \in \lambda(E)\}$ the maximum label assigned by λ. We define the *age* of a temporal graph $\lambda(G)$ as $\alpha(\lambda) = \lambda_{\max} - \lambda_{\min} + 1$. Note that in case $\lambda_{\min} = 1$ then we have $\alpha(\lambda) = \lambda_{\max}$. For every graph G we denote by $\mathcal{L}_G$ the set of all possible labelings λ of G. Furthermore, for every $k \in \mathbb{N}$, we define $\mathcal{L}_{G,k} = \{\lambda \in \mathcal{L}_G : \alpha(\lambda) \leq k\}$.

2.2 Further Definitions

For every time $r \in \mathbb{N}$, we define the *rth instance of a temporal graph $\lambda(G)$* as the static graph $\lambda(G, r) = (V, E(r))$, where $E(r) = \{e \in E : r \in \lambda(e)\}$ is the (possibly empty) set of all edges of the underlying graph G that are assigned label r by labeling λ. A temporal graph $\lambda(G)$ may be also viewed as a *sequence of static graphs* $(G_1, G_2, \ldots, G_{\alpha(\lambda)})$, where $G_i = \lambda(G, \lambda_{\min} + i - 1)$ for all $1 \leq i \leq \alpha(\lambda)$. Another, often convenient, representation of a temporal graph is the following.

Definition 2 *The* static expansion [4] *of a temporal graph $\lambda(G)$ is a* static digraph $H = (S, A)$*, and in particular a DAG, defined as follows. If $V = \{u_1, u_2, \ldots, u_n\}$ then $S = \{u_{ij} : \lambda_{\min} - 1 \leq i \leq$*

[1] The reason that we do not consider only digraphs and then allow undirected graphs to result as their special case, is that in that way an undirected edge would formally consist of two antiparallel edges. This would allow those edges to be labeled differently, unless we introduced an additional constraint preventing it. We've chosen to avoid this by considering explicit undirected graphs (whenever required) with at most one bidirectional edge per pair of nodes.

[2] The reader may be wondering whether it is pointless to allow the assignment of no labels to an edge e of G, as it would have been equivalent to delete e from G in the first place. Even though this is true for temporal graphs provided as input, it isn't for temporal graphs that will be *designed* by an algorithm based on an underlying graph. In the latter case, it is the algorithm's task to decide whether some of the provided edges need not be ever made available.

[3] Even though both names are almost equally used in the literature, in this paper we have chosen to use the term "temporal" in order to avoid confusion of readers that are more familiar with the use of the term "dynamic" to refer to dynamically updated instances, with which usually an algorithm has to deal in an online way (including the rich literature of problems in which the algorithm has to maintain a graph property that is being disturbed by adversarial graph modifications).

[4] The notion of static expansion is related to the notion of *time-expanded graphs* of temporal graphs such as periodic, or resulting from public transportation networks (cf. [24, 27]).

$\lambda_{\max}, 1 \leq j \leq n\}$ *and* $A = \{(u_{(i-1)j}, u_{ij'}) : \text{if } j = j' \text{ or } (u_j, u'_j) \in E(i) \text{ for some } \lambda_{\min} \leq i \leq \lambda_{\max}\}$. *In words, we create* $\alpha(\lambda) + 1$ *copies of* V *representing the nodes over time (time-nodes) and add outgoing edges from time-nodes of one level only to time-nodes of the next level. In particular, we connect a time-node* $u_{(i-1)j}$ *to its own subsequent copy* u_{ij} *and to every time node* $u_{ij'}$ *s.t.* (u_j, u'_j) *is an edge of* $\lambda(G)$ *at time* i.

A *journey* (or *time-respecting path*) J of a temporal graph $\lambda(G)$ is a path $(e_1, e_2, \ldots, e_k)$ of the underlying graph $G = (V, E)$, where $e_i \in E$, together with labels $l_1 < l_2 < \ldots < l_k$ such that $l_i \in \lambda(e_i)$ for all $1 \leq i \leq k$. In words, a journey is a path that uses strictly increasing edge-labels. If labeling λ defines a journey on some path P of G then we also say that λ *preserves* P. A natural notation for a journey is $(e_1, l_1), (e_2, l_2), \ldots, (e_k, l_k)$. We call each (e_i, l_i) a *time-edge* as it corresponds to the availability of edge e_i at some time l_i. We call l_1 the *departure time* and l_k the *arrival time* of journey J and denote them by $d(J)$ and $a(J)$, respectively. A (u, v)-journey J is called *foremost from time* t if $d(J) \geq t$ and $a(J)$ is minimized. Formally, let $\mathcal{J}$ be the set of all (u, v)-journeys J with $d(J) \geq t$. A $J \in \mathcal{J}$ is foremost if $a(J) = \min_{J' \in \mathcal{J}}\{a(J')\}$. A journey J is called *fastest* if $a(J) - d(J) + 1$ is minimized. We call $a(J) - d(J) + 1$ the *duration* of the journey. A journey J is called *shortest* if k is minimized, that is it minimizes the number of nodes visited (also called number of hops).

We say that a journey J *leaves from node* u (*arrives at node* u, resp.) *at time* t if (u, v, t) ((v, u, t), resp.) is a time-edge of J. Two journeys are called *out-disjoint* (*in-disjoint*, respectively) if they never leave from (arrive at, resp.) the same node at the same time.

Given a set $\mathcal{J}$ of (s, v)-journeys we define their *arrival time* as $a(\mathcal{J}) = \max_{J \in \mathcal{J}} \{a(J)\}$. We say that a set $\mathcal{J}$ of (s, v)-journeys satisfying some constraint c (e.g. containing at least k journeys and/or containing only out-disjoint journeys) is *foremost* if $a(\mathcal{J})$ is minimized over all sets of journeys satisfying the constraint.

If, in addition to the labeling λ, a positive weight $w(e) > 0$ is assigned to every edge $e \in E$, then we call a temporal graph a *weighted* temporal graph. In case of a weighted temporal graph, by "shortest journey" we mean a journey that minimizes the sum of the weights of its edges.

Throughout the text we denote by n the number of nodes and by m and m_t the number of edges of graphs and temporal graphs, respectively. In case of a temporal graph, by "number of edges" we mean "number of time-edges", i.e. $m_t = |\lambda|$. By $d(G)$ we denote the diameter of a (di)graph G, that is the length of the longest shortest path between any two nodes of G. By δ_u we denote the degree of a node $u \in V(G)$ (in case of an undirected graph G).

Part I

3 Journey Problems

3.1 Foremost Journeys

We are given (in its full "offline" description) a *temporal graph* $\lambda(G)$, where $G = (V, E)$, a distinguished source node $s \in V$, and a time $\lambda_{\min} \leq t_{start} \leq \lambda_{\max}$ and we are asked for all $w \in V \backslash \{s\}$ to compute a foremost (s, w)-journey from time t_{start}.

Theorem 1 *Algorithm 1 correctly computes for all* $w \in V \backslash \{s\}$ *a foremost* (s, w)*-journey from time* t_{start}. *The running time of the algorithm is* $O(n\lambda_{\max} + m_t)$.

Proof. Assume that at the end of round $t-1$ all nodes in R have been reached by foremost journeys from s. Let (u, v, t) be a time-edge s.t. $u \in R$ and $v \notin R$ and let $f(s, u)$ denote the foremost journey from s to u. We claim that $J = f(s, u), (u, v, t)$ is a foremost journey from s to v. Recall that we denote the arrival time of J by $a(J)$. To see that our claim holds assume that there is some other journey J' s.t. $a(J') < a(J)$. So there must be some time-edge (w, z, t') for $w \in R$, $z \notin R$ and

Algorithm 1 FJ

Input: Temporal graph $\lambda(G)$ (full "offline" description), source node $s \in V$, and time t_{start}, where $\lambda_{\min} \leq t_{start} \leq \lambda_{\max}$. The input is represented by an array A_v with $\lambda_{\max} - \lambda_{\min} + 1$ entries for every node v, where the entry $A_v[t]$ stores a pointer to the linked list of the adjacent nodes of v at time step t.

Output: For all $v \in V \backslash \{s\}$ a foremost (s, v)-journey from time t_{start}. In particular, outputs for every v a pair $(p[v], a[v])$, where $p[v]$ is the predecessor node of v on the journey and $a[v]$ is the arrival time of the journey at v (the pair as a whole may be viewed as the predecessor time-node of v on the journey).

```
1:  R ← {s}, t ← t_start
2:  for each v ∈ V\{s} do
3:      p[v] ← ∅
4:      a[v] ← ∞
5:  while R ≠ V and t ≠ λ_max + 1 do
6:      C ← ∅
7:      for each u ∈ R do
8:          for each (u, v) ∈ E(t) do
9:              if p[v] = ∅ then {that is, v ∉ R}
10:                 p[v] ← u
11:                 a[v] ← t
12:                 C ← C ∪ {v}
13:     R ← R ∪ C
14:     t++
```

$t' < t$. However, this contradicts the fact that $z \notin R$ as the algorithm should have added it in R at time t'. The proof follows by induction on t beginning from $t = t_{start}$ at which time $R = \{s\}$ (s has trivially been reached by a foremost journey from itself so the claim holds for the base case).

We now prove that the time complexity of the algorithm is $O(n\lambda_{\max} + m_t)$. In the worst-case, the last node may be inserted at step $\lambda_{\max}$, so the while loop is executed $O(\lambda_{\max})$ times. In each execution of the while loop, the algorithm visits the $O(n)$ nodes of the current set R in the worst-case (e.g. when all nodes but one have been added into R from the first step). For each such node v and for each time $\lambda_{\min} \leq t \leq \lambda_{\max}$ the algorithm first locates the entry $A_v[t]$ in the array A_v in constant time and then it visits the whole linked list of the adjacent nodes of v at time step t. All these operations can be performed in $O(n\lambda_{\max} + m_t)$ time in total. ■

3.2 Shortest Journeys with Weights

Theorem 2 *Let $\lambda(G)$, where $G = (V, E)$, be a weighted temporal graph with n vertices and m edges. Assume also that $|\lambda(e)| = 1$ for all $e \in E$, i.e. there is a single label on each edge (this implies also that $m_t = m$). Let $s, t \in V$. Then, we can compute a shortest journey J between s and t in $\lambda(G)$ (or report that no such journey exists) in $O(m \log m + \sum_{v \in V} \delta_v^2) = O(n^3)$ time, where δ_v is the degree of v in $\lambda(G)$.*

Proof. First, we may assume without loss of generality that $\lambda(G)$ is a connected graph, and thus $m \geq n - 1$. For the purposes of the proof we construct from $\lambda(G)$ a weighted directed graph H with two specific vertices s', t', such that there exists a journey J in $\lambda(G)$ between s and t if and only if there is a directed path P in H from s' to t'. Furthermore, if such paths exist, then the weight of the shortest journey J of $\lambda(G)$ between s and t equals the weight of the shortest directed path P of H from s' to t'.

First consider the (undirected) graph G' that we obtain when we add two vertices s_0 and t_0 to $\lambda(G)$ and the edges $s_0 s$ and $t t_0$. Assign to these two new edges the weight zero and assign to them the time labels $\lambda(s_0 s) = 0$ and $\lambda(t t_0) = \lambda_{\max} + 1$. Then, clearly there exists a time-respecting

path between s and t in $\lambda(G)$ if and only if there exists a time-respecting path between s_0 and t_0 in G', while the weights of these two paths coincide. For simplicity of the presentation, denote in the following by V and E the vertex and edge sets of G', respectively. Then we construct $H = (V_H, E_H)$ from $G' = (V, E)$ as follows. Let $V_H = E$. Furthermore, for every vertex $v \in V$, denote by $M(v) = \{vu : u \in N(v)\}$ the set of all incident edges to v in G'. For every pair $e_1, e_2 \in M(v)$ for some $v \in V$, add the arc $\widehat{e_1e_2}$ to E_H if and only if $\lambda(e_1) < \lambda(e_2)$. In this case, we assign to the arc $\widehat{e_1e_2}$ of E_H the weight $w_H(\widehat{e_1e_2}) = w(e_2)$.

Suppose first that G' has a journey between s_0 and t_0. Let $J = (u_0, u_1, \ldots, u_k)$, where $u_0 = s_0$ and $u_k = t_0$, be the shortest among them with respect to the weight function w of G'. Then, by the definition of G', s_0s and tt_0 are the first and the last edges of J. Furthermore, by the definition of a time-respecting path, $\lambda(u_{i-1}u_i) < \lambda(u_iu_{i+1})$ for every $i = 1, 2, \ldots, k-1$. Therefore, by the above construction of H, there exists the directed path $Q = (e_0, e_1, \ldots, e_{k-1})$ in H, where $e_i = u_iu_{i+1}$ for every $i = 0, 1, \ldots, k-1$. Note that $e_0 = s_0s$ and that $e_{k-1} = tt_0$. Furthermore, in the weight function w_H of H, $w_H(\widehat{e_ie_{i+1}}) = w(e_{i+1})$ for every $i = 0, 1, \ldots, k-2$. Note that $w_H(\widehat{e_{k-2}e_{k-1}}) = w(e_{k-1}) = w(u_{k-1}u_k)$, i.e. $w_H(\widehat{e_{k-2}e_{k-1}}) = w(tt_0) = 0$. Thus, the total weight $w(J)$ of J in G' equals the total weight $w_H(Q)$ of Q in H.

Let now $s_H = s_0s$ and $t_H = tt_0$. Suppose now that H has a path between s_H and t_H. Let $Q = (e_0, e_1, \ldots, e_k)$, where $e_0 = s_H$ and $e_k = t_H$, be the shortest among them with respect to the weight function w_H of H. Since Q is a directed path between s_H and t_H, $\lambda(e_i) < \lambda(e_{i+1})$ for every $i = 0, 1, \ldots, k-1$ by the construction of H. Furthermore, the edges e_i and e_{i+1} of G' are incident for every $i = 0, 1, \ldots, k-1$. Denote now by p_i the common vertex of the edges e_i and e_{i+1} in G' for every $i = 0, 1, \ldots, k-1$. We will prove that $p_i \neq p_{i+1}$ for every $i = 0, 1, \ldots, k-2$. Suppose otherwise that $p_i = p_{i+1}$ for some $0 \leq i \leq k-2$. Then the edges e_i, e_{i+1}, and e_{i+2} of G' are as it is shown in Figure 2, where $e_i = ad$, $e_{i+1} = bd$, $e_{i+2} = cd$, and $d = p_i = p_{i+1}$ is the common point of the edges e_i, e_{i+1}, and e_{i+2}. However, since $\lambda(e_i) < \lambda(e_{i+1})$ and $\lambda(e_{i+1}) < \lambda(e_{i+2})$, it follows that $\lambda(e_i) < \lambda(e_{i+2})$, and thus there exists the arc $\widehat{e_ie_{i+2}}$ in the directed graph H. Furthermore $w_H(\widehat{e_ie_{i+2}}) = w_H(\widehat{e_{i+1}e_{i+2}}) = w(e_{i+2})$, and thus $w_H(\widehat{e_ie_{i+1}}) + w_H(\widehat{e_{i+1}e_{i+2}}) > w_H(\widehat{e_ie_{i+2}})$. Therefore there exists in H the strictly shorter directed path $Q' = (e_0, e_1, \ldots, e_i, e_{i+2} \ldots, e_k)$ between $e_0 = s_H$ and $e_k = t_H$. This is a contradiction, since Q is the shortest directed path between s_H and t_H. Therefore $p_i \neq p_{i+1}$ for every $i = 0, 1, \ldots, k-2$. Thus, we can denote now $e_i = p_{i-1}p_i$ for every $i = 1, 2, \ldots, k$, where $p_0 = s_0$ and $p_k = t_0$. That is, $J = (p_0, p_1, \ldots, p_{k+1})$ is a walk in G' between $p_0 = s_0$ and $p_k = t_0$.

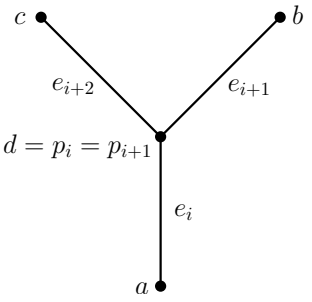

Figure 2: A forbidden configuration.

Since Q is a simple directed path, it follows that every edge of J appears exactly once in J, and thus J is a path of G'. Now we will prove that J is actually a simple path of G'. Suppose otherwise that $p_i = p_j$ for some $0 \leq i < j \leq k+1$. If $p_j = p_k$, i.e. $p_j = t_0$, then the subpath $(p_0, p_1, \ldots, p_i)$ of J implies a strictly shorter directed path Q' than Q between s_H and t_H in H, which is a contradiction. Therefore $p_j \neq p_k$. Then, since $\lambda(p_{i-1}p_i) < \lambda(p_ip_{i+1})$ for every $i = 0, 1, \ldots, k-1$ by the construction of the directed graph H, it follows in particular that $\lambda(p_{i-1}p_i) < \lambda(p_jp_{j+1})$, and thus $\widehat{e_ie_{j+1}}$ is an arc in the directed graph H. Thus the path $(p_0, p_1, \ldots, p_i, p_{j+1}, \ldots, p_k)$ of G' implies a strictly shorter directed path Q' than Q between s_H and t_H in H, which is again a contradiction. Therefore $p_i \neq p_j$ for every $0 \leq i < j \leq k+1$ in J, and thus J is a simple path in G' between $p_0 = s_0$ and $p_k = t_0$. Finally, it is easy to check that the weight $w(J)$ of J in G' equals

the weight $w_H(Q)$ of Q in H.

Summarizing, there exists a journey J in G' between s_0 and t_0 if and only if there is a directed path Q in H from s_H to t_H. Furthermore, if such paths exist, then the weight of the shortest journey J of G' between s_0 and t_0 equals the weight of the shortest directed path Q of H from s_H to t_H.

Moreover, the above proof immediately implies an efficient algorithm for computing the graph H from $\lambda(G)$ (by first constructing the auxiliary graph G' from $\lambda(G)$). This can be done in $O(\sum_{v\in V}\delta_v^2)$ time. Indeed, for every vertex v of G' we add at most $2\binom{\delta_v}{2} = \delta_v(\delta_v - 1)$ arcs to H. That is, $|V_H| = m + 2$ and $|E_H| \leq \sum_{v\in V(G')}\delta_v(\delta_v - 1) = O(\sum_{v\in V}\delta_v^2)$. After we construct H, we can compute a shortest directed path between s_H and t_H in $O(|E_H| + |V_H|\log|V_H|)$ time using Dijkstra's algorithm with Fibonacci heaps [12]. That is, we can compute a shortest directed path Q in H between s_H and t_H in $O(m\log m + \sum_{v\in V}\delta_v^2)$ time. Once we have computed the path Q, we can easily construct the shortest undirected journey J in $\lambda(G)$ between s and t in $O(m+n)$ time. This completes the proof of the theorem. ■

4 A Menger's Analogue for Temporal Graphs

In [15], Kempe *et al.* proved that Menger's theorem, at least in its original formulation, does not hold for single-label temporal networks in which journeys must have non-decreasing labels (and not necessarily strictly increasing as in our case). For a counterexample, it is not hard to see in Figure 3 that there are no two disjoint time-respecting paths from v_1 to v_4 but after deleting any one node (other than v_1 or v_4) there still remains a time-respecting v_1-v_4 path. Moreover, they proved that the violation of Menger's theorem in such temporal networks renders the computation of the number of disjoint s-t paths NP-complete.

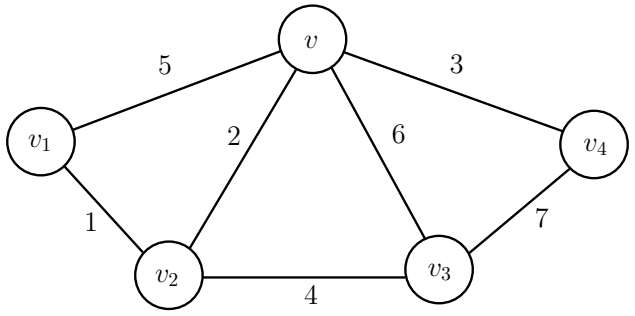

Figure 3: A counterexample of Menger's theorem for temporal networks (adopted from [15]). Each edge has a single time-label indicating its availability time.

We prove in this section that, in contrast to the above important negative result, there is a natural analogue of Menger's theorem that is valid for all temporal networks. In Theorem 3, we define this analogue and prove its validity. Then as an illustration (§4.1), we show how using our theorem can simplify the proof of a recent token dissemination result.

When we say that we remove *node departure time* (u, t) we mean that we remove *all time-edges leaving u at time t*, i.e. we remove label t from all (u, v) edges (for all $v \in V$). In case of an undirected graph, we replace each edge by two antiparallel edges and remove label t only from the outgoing edges of u. So, when we ask how many node departure times are needed to separate two nodes s and v we mean how many node departure times must be selected so that after the removal of all the corresponding time-edges the resulting temporal graph has no (s, v)-journey. [5]

[5]Note that this is a different question from how many time-edges must be removed and, as we shall see, the latter question does not result in a Menger's analogue. Of course, removing a node departure time again results in the removal of some time-edges, but a Menger's analogue based on the number of those edges would not work. Instead, what turns out to work is an analogue based on counting the number of node departure times.

Theorem 3 (Menger's Temporal Analogue) *Take any temporal graph $\lambda(G)$, where $G = (V, E)$, with two distinguished nodes s and v. The maximum number of out-disjoint journeys from s to v is equal to the minimum number of node departure times needed to separate s from v.*

Proof. Assume, in order to simplify notation, that $\lambda_{\min} = 1$. Take the static expansion $H = (S, A)$ of $\lambda(G)$. Let $\{u_{i1}\}$ and $\{u_{in}\}$ represent s and v over time, respectively (first and last columns, respectively), where $0 \leq i \leq \lambda_{\max}$. We extend H as follows. For each u_{ij}, $0 \leq i \leq \lambda_{\max} - 1$, with at least 2 outgoing edges to nodes different than $u_{(i+1)j}$, e.g. to nodes $u_{(i+1)j_1}, u_{(i+1)j_2}, \ldots, u_{(i+1)j_k}$, we add a new node w_{ij} and the edges (u_{ij}, w_{ij}) and $(w_{ij}, u_{(i+1)j_1}), (w_{ij}, u_{(i+1)j_2}), \ldots, (w_{ij}, u_{(i+1)j_k})$. We also define an edge capacity function $c : A \to \{1, \lambda_{\max}\}$ as follows. All edges of the form $(u_{ij}, u_{(i+1)j})$ take capacity $\lambda_{\max}$ and all other edges take capacity 1. We are interested in the maximum flow from u_{01} to $u_{\lambda_{\max} n}$. As this is simply a usual static flow network, the max-flow min-cut theorem applies stating that the maximum flow from u_{01} to $u_{\lambda_{\max} n}$ is equal to the minimum of the capacity of a cut separating u_{01} from $u_{\lambda_{\max} n}$. So it suffices to show that (i) the maximum number of out-disjoint journeys from s to v is equal to the maximum flow from u_{01} to $u_{\lambda_{\max} n}$ and (ii) the minimum number of node departure times needed to separate s from v is equal to the minimum of the capacity of a cut separating u_{01} from $u_{\lambda_{\max} n}$.

For (i) observe that any set of h out-disjoint journeys from s to v corresponds to a set of h disjoint paths from u_{01} to $u_{\lambda_{\max} n}$ w.r.t. diagonal edges (edges in $E \backslash \{(u_{ij}, u_{(i+1)j})\}$) and inversely, so their maximums are equal. Next observe that any set of h disjoint paths from u_{01} to $u_{\lambda_{\max} n}$ w.r.t. diagonal edges corresponds to an integral u_{01}-$u_{\lambda_{\max} n}$ flow on H of value h and inversely. As the maximum integral u_{01}-$u_{\lambda_{\max} n}$ flow is equal to the maximum u_{01}-$u_{\lambda_{\max} n}$ flow (the capacities are integral and thus the integrality theorem of maximum flows applies) we conclude that the maximum u_{01}-$u_{\lambda_{\max} n}$ flow is equal to the maximum number of out-disjoint journeys from s to v.

For (ii) observe that any set of r node departure times that separate s from v corresponds to a set of r diagonal edges leaving u_{ij} nodes (ending either in w_{ij} or in $u_{(i+1)j'}$ nodes) that separate u_{01} from $u_{\lambda_{\max} n}$ and inversely. Finally, observe that there is a minimum u_{01}-$u_{\lambda_{\max} n}$ cut on H that only uses such edges: for if a minimum cut uses vertical edges we can replace them by diagonal edges and we can replace all edges leaving a w_{ij} node by the edge (u_{ij}, w_{ij}) without increasing the total capacity. ■

Corollary 1 *By symmetry we have that the maximum number of in-disjoint journeys from s to v is equal to the minimum number of node arrival times needed to separate s from v.*

Corollary 2 *The following alternative statements are both valid:*

- *The maximum number of time-node disjoint journeys from s to v is equal to the minimum number of time-nodes needed to separate s from v.*
- *The maximum number of time-edge disjoint journeys from s to v is equal to the minimum number of time-edges needed to separate s from v.* [6]

The following version is though violated: "the maximum number of out-disjoint (or in-disjoint) journeys from s to v is equal to the minimum number of time-edges needed to separate s from v" (see Figure 4). The same holds for the original statement of Menger's theorem as discussed in the beginning of this section (see [15]).

4.1 An Application: Foremost Dissemination (Journey Packing)

Consider the following problem. We are given a temporal graph $\lambda(G)$, where $G = (V, E)$, a source node s, a sink node v and an integer q. We are asked to find the minimum arrival time of a set of q out-disjoint (s, v)-journeys or even the minimizing set itself.

[6] By time-node disjointness we mean that they do not meet on the same node at the same time (in terms of the expansion graph the corresponding paths should be disjoint in the classical sense) and by time-edge disjointness that they do not use the same time-edge (which again translates to using the same diagonal edge on the expansion graph).

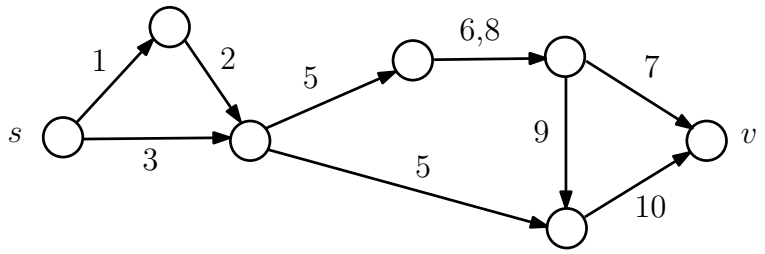

Figure 4: A violation of an invalid Menger's analogue. Both edges labeled 5 must be removed to separate s from v however there are no two out-disjoint journeys from s to v (all (s, v)-journeys must use some edge labeled 5).

By exploiting the Menger's analogue proved in Theorem 3 (and in order to provide an example application of it), we give an alternative (and probably simpler to appreciate) proof of the following Lemma from [10] (stated as Lemma 1 below) holding for a special case of temporal networks, namely those that have *connected instances*. Formally, a temporal network $\lambda(G)$ is said to have connected instances if $\lambda(G, t)$ is connected at all times $t \in \mathbb{N}$. The problem under consideration is distributed k-token dissemination: there are k tokens assigned to some given source nodes. In each round (i.e. discrete moment in the temporal network), each node selects a single token to be sent to all of its current neighbors (i.e. broadcast). The current neighbors at round i are those defined by $E(i)$. The goal of a distributed protocol (or of a centralized strategy for the same problem) is to deliver all tokens to a given sink node v as fast as possible. We assume that the algorithms know the temporal network in advance.

Lemma 1 *Let there be $k \leq n$ tokens at given source nodes and let v be an arbitrary node. Then, all the tokens can be sent to v using broadcasts in $O(n)$ rounds.*

Let $S = \{s_1, s_2, \ldots, s_h\}$ be the set of source nodes and let $k(s_i)$ be the number of tokens of source node s_i, so that $\sum_{1 \leq i \leq h} k(s_i) = k$. Clearly, it suffices to prove the following lemma.

Lemma 2 *We are given a temporal graph $\lambda(G)$ with connected instances and age $\alpha(\lambda) = n + k$. We are also given a set of source nodes $S \subseteq V$, a mapping $k : S \to \mathbb{N}_{\geq 1}$ so that $\sum_{s \in S} k(s) = k$, and a sink node v. Then there are at least k out-disjoint journeys from S to v such that $k(s_i)$ journeys leave from each source node s_i.*

Proof. We conceive $k(s)$ as the number of tokens of source s. Number the tokens arbitrarily. Create a supersource node s' and connect it to the source node with token i by an edge labeled i. Increase all other edge labels by k. Clearly the new temporal graph $D = \lambda'(G')$ has asymptotically the same age as the original and all properties have been preserved (we just shifted the original temporal graph in the time dimension). Moreover, if there are k out-disjoint journeys from s' to v in D then by construction of the edges leaving s' we have that precisely $k(s)$ of these journeys must be leaving from each source $s \in S$. So it suffices to show that there are k out-disjoint journeys from s' to v. By Theorem 3 it is equivalent to show that the minimum number of departure times that must be removed from D to separate s' from v is k. Assume that we remove $y < k$ departure times. Then for more than n rounds all departure times are available (as we have $n + 2k$ rounds and we just have $y < k$ removals). As every instance of G is connected, we have that there is always an edge in the cut between the nodes that have been reached by s' already and those that have not, unless we remove some departure times. As for more than n rounds all departure times are available it is immediate to observe that s' reaches v implying that we cannot separate s' from v with less that k removals and this completes the proof. ■

Part II

5 Minimum Cost Temporal Connectivity

In this section, we introduce some cost measures for maintaining different types of temporal connectivity. According to these temporal connectivity types, individuals are required to be capable to communicate with other individuals over the dynamic network, possibly with further restrictions on the timing of these connections. We initiate this study by considering the following fundamental problem: Given a (di)graph G, assign labels to the edges of G so that the resulting temporal graph $\lambda(G)$ minimizes some parameter and at the same time preserves some connectivity property of G in the time dimension. For a simple illustration of this, consider the case in which $\lambda(G)$ should contain a journey from u to v if and only if there exists a path from u to v in G. In this example, the reachabilities of G completely define the temporal reachabilities that $\lambda(G)$ is required to have.

We consider two cost optimization criteria for a (di)graph G. The first one, called *temporality* of G, measures the maximum number of labels that an edge of G has been assigned. The second one, called *temporal cost* of G, measures the total number of labels that have been assigned to all edges of G. That is, if we interpret the number of assigned labels as a measure of *cost*, the temporality (resp. the temporal cost) of G is a measure of the decentralized (resp. centralized) cost of the network, where only the cost of individual edges (resp. the total cost over all edges) is considered. We introduce these cost parameters in Definition 3. Each of these two cost measures can be minimized subject to some particular connectivity property $\mathcal{P}$ that the labeled graph $\lambda(G)$ has to satisfy. For simplicity of notation, we consider in Definition 3 the connectivity property $\mathcal{P}$ as a subset of the set $\mathcal{L}_G$ of all possible labelings λ on the (di)graph G. Furthermore, the minimization of each of these two cost measures can be affected by some problem-specific constraints on the labels that we are allowed to use. We consider here one of the most natural constraints, namely an upper bound on the *age* of the constructed labeling λ.

Definition 3 *Let $G = (V, E)$ be a (di)graph, $\alpha_{\max} \in \mathbb{N}$, and $\mathcal{P}$ be a connectivity property. Then the* temporality *of $(G, \mathcal{P}, \alpha_{\max})$ is*

$$\tau(G, \mathcal{P}, \alpha_{\max}) = \min_{\lambda \in \mathcal{P} \cap \mathcal{L}_{G,\alpha_{\max}}} \max_{e \in E} |\lambda(e)|$$

and the temporal cost *of $(G, \mathcal{P}, \alpha_{\max})$ is*

$$\kappa(G, \mathcal{P}, \alpha_{\max}) = \min_{\lambda \in \mathcal{P} \cap \mathcal{L}_{G,\alpha_{\max}}} \sum_{e \in E} |\lambda(e)|$$

Furthermore $\tau(G, \mathcal{P}) = \tau(G, \mathcal{P}, \infty)$ and $\kappa(G, \mathcal{P}) = \kappa(G, \mathcal{P}, \infty)$.

Note that Definition 3 can be stated for an arbitrary property $\mathcal{P}$ of the labeled graph $\lambda(G)$ (e.g. some proper coloring-preserving property). Nevertheless, we only consider here $\mathcal{P}$ to be a connectivity property of $\lambda(G)$. In particular, we investigate the following two connectivity properties $\mathcal{P}$:

- *all-paths*$(G) = \{\lambda \in \mathcal{L}_G$: for all simple paths P of G, λ preserves $P\}$,
- *reach*$(G) = \{\lambda \in \mathcal{L}_G$: for all $u, v \in V$ where v is reachable from u in G, λ preserves at least one simple path from u to $v\}$.

5.1 Basic Properties of Temporality Parameters

5.1.1 Preserving All Paths

We begin with some simple observations on $\tau(G, \textit{all paths})$. Recall that given a (di)graph G our goal is to label G so that all simple paths of G are preserved by using as few labels per edge as

possible. From now on, when we say "graph" we will mean a directed one and we will state it explicitly when our focus is on undirected graphs.

Another interesting observation is that if $p(G)$ is the length of the longest path in G then we can trivially preserve all paths of G by using $p(G)$ labels per edge. Give to every edge the labels $\{1, 2, \ldots, p(G)\}$ and observe that for every path $e_1, e_2, \ldots, e_k$ of G we can use the increasing sequence of labels $1, 2, \ldots, k$ due to the fact that $k \leq p(G)$. Thus, we conclude that the upper bound $\tau(G, \textit{all paths}) \leq p(G)$ holds for all graphs G. Of course, note that equality is easily violated. For example, a directed line has $p(G) = n$ but $\tau(G, \textit{all paths}) = 1$.

Observation 1 *$\tau(G, \textit{all paths}) \leq p(G)$ for all graphs G.*

Directed Rings. The following proposition states that if G is a directed ring then the temporality of preserving all paths is 2. This means that the minimum number of labels per edge that preserve all simple paths of a ring is 2. As the proof was already sketched in Section 1, we don't provide a proof here.

Proposition 1 *$\tau(G, \textit{all paths}) = 2$ when G is a ring and $\tau(G, \textit{all paths}) \geq 2$ when G contains a ring.*

Directed Acyclic Graphs. A topological sort of a digraph G is a linear ordering of its nodes such that if G contains an edge (u, v) then u appears before v in the ordering. It is well known that a digraph G can be topologically sorted iff it has no directed cycles that is iff it is a DAG. A topological sort of a graph can be seen as placing the nodes on a horizontal line in such a way that all edges go from left to right; see e.g. [8, page 549].

Proposition 2 *If G is a DAG then $\tau(G, \textit{all paths}) = 1$.*

Proof. Take a topological sort $u_1, u_2, \ldots, u_n$ of G. Clearly, every edge is of the form (u_i, u_j) where $i < j$. Give to every edge (u_i, u_j) label i, that is $\lambda(u_i, u_j) = i$ for all $(u_i, u_j) \in E$. Now take any node u_l. Each of its incoming edges has some label $l' < l$ and all its outgoing edges have label l. Now take any simple path $p = v_1, v_2, \ldots, v_k$ of G. Clearly, v_i appears before v_{i+1} in the topological sort for all $1 \leq i \leq k-1$, which implies that $\lambda(v_i, v_{i+1}) < \lambda(v_{i+1}, v_{i+2})$, for all $1 \leq i \leq k-2$. This proves that p is preserved. As we have preserved all simple paths with a single label on every edge, we conclude that $\tau(G, \textit{all paths}) = 1$ as required. ∎

5.1.2 Preserving All Reachabilities

Now, instead of preserving all paths, we impose the apparently simpler requirement of preserving just a single path between every reachability pair $u, v \in V$. We claim that it is sufficient to understand how $\tau(G, reach)$, behaves on strongly connected digraphs. Let $\mathcal{C}(G)$ be the set of all strongly connected components of a digraph G. The following lemma proves that, w.r.t. the *reach* property, the temporality of any digraph G is equal to the maximum temporality of its components.

Lemma 3 *$\tau(G, reach) = \max\{1, \max_{C \in \mathcal{C}(G)} \tau(C, reach)\}$ for every digraph G with at least one edge. In the case of no edge, $\tau(G, reach) = 0$ trivially.*

Proof. Take any digraph G. Now take the DAG D of the strongly connected components of G. The nodes of D are the components of G and there is an edge from component C to component C' if there is an edge in G from some node of C to some node of C'. As D is a DAG, we can obtain a topological sort of it which is a labeling $C_1, C_2, \ldots, C_t$ of the t components so that all edges between components go only from left to right.

In the case where at least one component has at least 2 nodes (in which case $\max_{C \in \mathcal{C}(G)} \tau(C, reach) \geq 1$), we have to prove that we can label G by using at most $\max_{1 \leq i \leq t} \tau(C_i, reach)$ labels per edge and that we cannot do better than this. Consider the following labeling process. For each component C_i define $d_i = \min_{\lambda \in \mathcal{C}_i}(\lambda_{\max}(\lambda) - \lambda_{\min}(\lambda))$, where

$\mathcal{C}_i$ is the set of all labelings of C_i that preserve all of its reachabilities using at most $\tau(C_i, reach)$ labels per edge. Note that any C_i can be labeled beginning from any desirable $\lambda_{\min}$ with at most $\tau(C_i, reach)$ labels per edge and with λ_{max} equal to $\lambda_{\min} + d_i$. Now, label component C_1 with $\lambda_{\min} = 1$ and $\lambda_{\max} = 1 + d_1$. Label all edges leaving C_1 with label $d_1 + 2$. Label component C_2 with $\lambda_{\min} = d_1 + 3$ and $\lambda_{\max} = (d_1 + 3) + d_2$ and all its outgoing edges with label $(d_1 + 3) + d_2 + 1$. In general, label component C_i with $\lambda_{\min} = 1 + \sum_{1 \leq j \leq i-1}(d_j + 2)$ and $\lambda_{\max} = \lambda_{\min} + d_i$ and label all edges leaving C_i with label $\lambda_{\max} + 1$. It is not hard to see that this labeling scheme preserves all reachabilities of G using just one label on each edge of G corresponding to an edge of D and at most $\tau(C_i, reach)$ labels per edge inside each component C_i. Thus, it uses at most $\max_{1 \leq i \leq t} \tau(C_i, reach)$ labels on every edge. By observing that for each strongly connected component C_i, $\tau(C_i, reach)$ must be paid by any labeling of G that preserves all reachabilities in that component, the equality $\tau(G, reach) = \max_{C \in \mathcal{C}(G)} \tau(C, reach)$ follows.

In the extreme case where all components are just single nodes (in which case $\max_{C \in \mathcal{C}(G)} \tau(C, reach) = 0$), it holds that $D = G$, therefore G itself is a DAG and we only need 1 label per edge (as in Proposition 2) and, thus, $\tau(G, reach) = 1$. ■

Lemma 3 implies that any upper bound on the temporality of preserving the reachabilities of strongly connected digraphs can be used as an upper bound on the temporality of preserving the reachabilities of general digraphs. In view of this, we focus on strongly connected digraphs G.

We begin with a few simple but helpful observations. Obviously, $\tau(G, reach) \leq \tau(G, all\ paths)$ as any labeling that preserves all paths trivially preserves all reachabilities as well. If G is a clique then $\tau(G, reach) = 1$ as giving to each edge a single arbitrary label (e.g. label 1 to all) preserves all direct connections (one-step reachabilities) which are all present. If G is a directed ring (which is again strongly connected) then it is easy to see that $\tau(G, reach) = 2$. An interesting question is whether there is some bound on $\tau(G, reach)$ either for all digraphs or for specific families of digraphs. The following lemma proves that indeed there is a very satisfactory generic upper bound.

Lemma 4 *$\tau(G, reach) \leq 2$ for all strongly connected digraphs G.*

Proof. As G is strongly connected, if we pick any node u then for all v there is a (v, u) and a (u, v)-path. As for any v there is a (v, u)-path, then we may form an in-tree T_{in} rooted at u (that is a tree with all directions going upwards to u). Now beginning from the leaves give any direction preserving labeling (just begin from labels 1 at the leaves and increase them as you move upwards). Say that the depth is k which means that you have increased up to label k. Now consider an out-tree T_{out} rooted at u that has all edge directions going from u to the leaves. To make things simpler create second copies of all nodes but u so that the two trees are disjoint (w.r.t. to all nodes but u). In fact, one tree passes through all the first copies and arrives at u and the other tree begins from u and goes to all the second copies. Now we can begin the labeling of T_{out} from $k + 1$ increasing labels as we move away from u on T_{out}. This completes the construction.

Now take any two nodes w and v. Clearly, there is a time-respecting path from w to u and then a time-respecting path from u to v using greater labels so there is a time-respecting path from w to v. Finally, notice that for any edge on T_{in} there is at most one copy of that edge on T_{out} thus clearly we use at most 2 labels per edge. ■

Combining Lemma 3 and Lemma 4 gives the following theorem:

Theorem 4 *$\tau(G, reach) \leq 2$ for all digraphs G.*

5.1.3 Restricting the Age

Now notice that for all G we have $\tau(G, reach,\ d(G)) \leq d(G)$; recall that $d(G)$ denotes the diameter of (di)graph G. Indeed it suffices to label each edge by $\{1, 2, \ldots, d(G)\}$. Since every shortest path between two nodes has length at most $d(G)$, in this manner we preserve all shortest paths and thus all reachabilitities arriving always at most by time $d(G)$, thus we also preserve the diameter. Thus, a clique G has trivially $\tau(G, reach, d(G)) = 1$ as $d(G) = 1$ and

we can only have large $\tau(G, reach, d(G))$ in graphs with large diameter. For example, a directed ring G of size n has $\tau(G, reach, d(G)) = n - 1$ (note that on a ring it always holds that $\tau(G, reach, k) = \tau(G, all\ paths, k)$, as on a ring it happens that satisfying all reachabilities also satisfies all paths while the inverse is true for all graphs). Indeed, assume that from some edge e, label $1 \leq i \leq n-1$ is missing. It is easy to see that there is some shortest path between two nodes of the ring that in order to arrive by time $n-1$ must use edge e at time i. As this label is missing, it uses label $i+1$, thus it arrives by time n which is greater than the diameter. In this particular example we can preserve the diameter only if all edges have the labels $\{1, 2, \ldots, n-1\}$.

On the other hand, there are graphs with large diameter in which $\tau(G, reach, d(G))$ is small. This may also be the case even if G is strongly connected. For example, consider the graph with nodes $u_1, u_2, \ldots, u_n$ and edges (u_i, u_{i+1}) and (u_{i+1}, u_i) for all $1 \leq i \leq n-1$. In words, we have a directed line from u_1 to u_n and an inverse one from u_n to u_1. The diameter here is $n-1$ (e.g. the shortest path from u_1 to u_n). On the other hand, we have $\tau(G, reach, d(G)) = 1$: simply label one path $1, 2, ..., n-1$ and label the inverse one $1, 2, ..., n-1$ again, i.e. give to edges (u_i, u_{i+1}) and (u_{n-i+1}, u_{n-i+2}) label i. The reason here is that there are only two pairs of nodes that must necessarily use the long paths (u_1, u_n) and (u_n, u_1) and preserve the diameter $n-1$. All other smaller shortest paths between other pairs of nodes have now a big gap of $n-1$ to exploit.

We will now demonstrate what makes $\tau(G, reach, d(G))$ grow. It happens when many maximum shortest paths (those that determine the diameter of G) between different pairs of nodes that are additionally unique (the paths), in the sense that we must necessarily take them in order to preserve the reachabilities (it may hold even if they are not unique but this simplifies the argument), all pass through the same edge e but use e at many different times. It will be helpful to look at Figure 5. Each (u_i, v_i)-path is a unique shortest path between u_i and v_i and has additionally length equal to the diameter (i.e. it is also a maximum one), so we must necessarily preserve all 5 (u_i, v_i)-paths. Note now that each (u_i, v_i)-path passes through $e = (u_1, v_5)$ via its i-th edge. Each of these paths can only be preserved without violating $d(G)$ by assigning the labels $1, 2, \ldots, d(G)$, however note that then edge e must necessarily have all labels $1, 2, \ldots, d(G)$. To see this, notice simply that if any label i is missing from e then there is some maximum shortest path that goes through e at step i. As i is missing it cannot arrive sooner than time $d(G)+1$ which violates the preservation of the diameter.

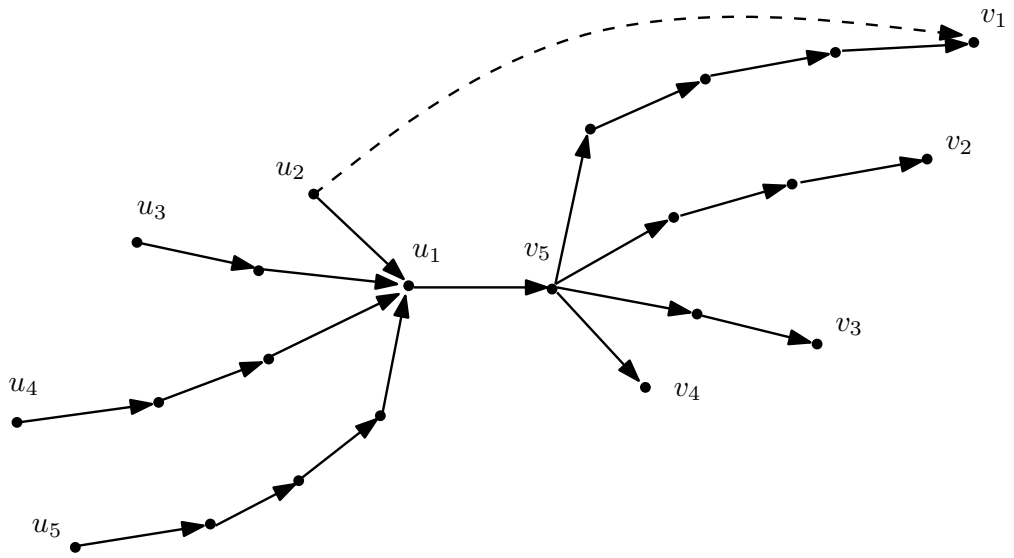

Figure 5: An example graph in which $\tau(G, reach, d(G)) = d(G)$. All paths longer than length 5 that are formed are not shortest paths, e.g. there is a path (the dashed one) of length at most 5 from u_2 to v_1 and the same for all other such pairs.

Undirected Tree. Now consider an undirected tree T.

Corollary 3 *If T is an undirected tree then $\tau(T, \text{all paths}, d(T)) \leq 2$.*

Proof. This follows as a simple corollary of Lemma 4. If we replace each undirected edge by two antiparallel edges, then T is a strongly connected digraph and, additionally, for every ordered pair of nodes (u, v) there is precisely one simple path from u to v. The latter implies that preserving all paths of T is equivalent to preserving all reachabilities of T. So, all assumptions of Lemma 4 are satisfied and therefore $\tau(T, \text{all paths}) \leq 2$. Finally, recall that the labeling of the construction in the proof of Lemma 4 starts increasing labels level-by-level from the leaves to the root and then from the root to the leaves, therefore the number of increments (i.e., the maximum label used) is upper bounded by the diameter of T, thus, $\tau(T, \text{all paths}, d(T)) \leq 2$ as required. ∎

Trade-off on a Ring. We shall now prove that there is a trade-off between the temporality and the age. In particular, we consider a directed ring $G = (e_1, e_2, \ldots, e_n)$, where the e_i are edges oriented clockwise. As we have already discussed, if $\alpha = n - 1$ then $\tau(G, \textit{all paths}, \alpha) = n - 1$ (which is the worst possible) and if $\alpha = 2(n - 1)$ then $\tau(G, \textit{all paths}, \alpha) = 2$ (which is the best possible). We now formalize the behavior of τ as α moves from $n - 1$ to $2(n - 1)$.

Theorem 5 *If G is a directed ring and $\alpha = (n - 1) + k$, where $1 \leq k \leq n - 1$, then $\tau(G, \textit{all paths}, \alpha) = \Theta(n/k)$ and in particular $\lfloor \frac{n-1}{k+1} \rfloor \leq \tau(G, \textit{all paths}, \alpha) \leq \lceil \frac{n}{k+1} \rceil + 1$. Moreover, $\tau(G, \textit{all paths}, n - 1) = n - 1$ (i.e. when $k = 0$).*

Proof. The proof of the upper bound is constructive. In particular, we present a labeling that preserves all paths of the ring G using at most $\lceil \frac{n}{k+1} \rceil + 1$ labels on every edge and maximum label $(n-1)+k$. Let the ring be $e_1, e_2, \ldots, e_n$ and clockwise. We say that an edge e_i is *satisfied* if there is a journey of length $n - 1$ beginning from e_i (clearly, considering only those journeys that do not use a label greater than $\alpha = (n - 1) + k$). Consider the following labeling procedure.

- For all $i = 0, 1, 2, \ldots, \lceil \frac{n}{k+1} \rceil - 2$
  - Assign label 1 to edge $e_{j=i(k+1)+1}$.
  - Beginning from edge e_{j+1}, assign labels $2, 3, \ldots, (n - 1) + k$ clockwise.
- For $i = \lceil \frac{n}{k+1} \rceil - 1$, assign label 1 to edge $e_{j=i(k+1)+1}$ and beginning from edge e_{j+1} assign labels $2, 3, \ldots, (n - 1) + (n - j)$ clockwise.

Note that in each iteration i we satisfy edges $e_{i(k+1)+1}, e_{i(k+1)+2}, \ldots, e_{(i+1)(k+1)}$, i.e. $k+1$ new edges, without leaving gaps. It follows that in $\lceil \frac{n}{k+1} \rceil$ iterations all edges have been satisfied. The first iteration assigns at most two labels on edge e_1 and every other iteration, apart from the last one, assigns one label on e_1 (and clearly at most one on every other edge), thus e_1 gets a total of at most $\lceil \frac{n}{k+1} \rceil + 1$ labels (and all other edges get at most this).

Now, for the lower bound, take an arbitrary edge, e.g. e_1. Given an edge e_i and a journey J from e_i to e_1 that uses label l_1 on e_1, define the delay of J as $l_1 - l(J)$, where $l(J)$ is the length of journey J i.e. $n - i + 2$. In words, the delay of a (e_i, e_1)-journey is the difference between the time at which the journey visits e_1 minus the fastest time that it could have visited e_1. Now, beginning from e_n count $k + 1$ times counterclockwise, i.e. consider edge e_{n-k}. We show that in order to satisfy e_{n-k} we must necessarily use one of the labels $\{k + 2, k + 3, \ldots, 2k + 2\}$ on e_1. To this end, notice that the delay of any journey that satisfies some edge can be at most k, the reason being that a delay of $k + 1$ or greater implies that the journey cannot visit $n - 1$ edges in less than $(n - 1) + (k + 1)$ time, thus it will have to use some label greater than $\alpha = (n - 1) + k$, which is the maximum allowed. Thus, the maximum label by which a journey that satisfies e_{n-k} can go through e_1 is $l(e_{n-k}) + k = 2k + 2$, where $l(e_i)$ denotes the length of the path beginning from the tail of e_i and ending at the head of e_1. Moreover, the minimum label by which any journey from e_{n-k} can go through e_1 is $l(e_{n-k}) = k + 2$. Thus, we conclude that any journey that satisfies e_{n-k} has to use one of the labels $\{k + 2, k + 3, \ldots, 2k + 2\}$ on e_1.

It is not hard to see that the above idea generalizes as follows. For all $i = 0, 1, \ldots, \lfloor \frac{n-1}{k+1} \rfloor - 1$, in order to satisfy edge $e_{n-i(k+1)+1}$ (note that $e_{n+1} = e_1$) we must necessarily use one of the labels $\{i(k+1)+1, i(k+1)+2, \ldots, (i+1)(k+1)\}$ on e_1. For example, for $i = 0$ we get $\{1, 2, \ldots, k+1\}$, for $i = 1$ we get $\{k+2, \ldots, 2k+2\}$, for $i = 2$ we get $\{2k+3, \ldots, 3k+3\}$, and so on. In summary, as the above sets are disjoint, if we begin from e_1 and move counterclockwise then for every $k+1$ edges we encounter we must pay for another (new) label on e_1 thus we pay at least $\lfloor \frac{n-1}{k+1} \rfloor$. ■

5.2 A Generic Method for Computing Lower Bounds for Temporality

Proposition 1 showed that graphs with directed cycles need at least 2 labels on some edge(s) in order for all paths to be preserved. Now a natural question to ask is whether we can preserve all paths of any graph by using at most 2 labels (i.e. whether $\tau(G, \textit{all paths}) \leq 2$ holds for all graphs). We shall prove that there are graphs G for which $\tau(G, \textit{all paths}) = \Omega(p(G))$ (recall that $p(G)$ denotes the length of the longest path in G), that is graphs in which the optimum labeling, w.r.t. temporality, is very close to the trivial labeling $\lambda(e) = \{1, 2, \ldots, p(G)\}$, for all $e \in E$, that always preserves all paths.

Definition 4 *Call a set $K = \{e_1, e_2, \ldots, e_k\} \subseteq E(G)$ of edges of a digraph G an* edge-kernel *if for every permutation $\pi = (e_{i_1}, e_{i_2}, \ldots, e_{i_k})$ of the elements of K there is a simple path P of G that visits all edges of K in the ordering defined by the permutation π.*

We will now prove that an edge-kernel of size k needs at least k labels on some edges. Our proof is constructive. In particular, given any labeling using $k-1$ labels on an edge-kernel of size k, we present a specific path that forces a kth label to appear.

Theorem 6 (Edge-kernel Lower Bound) *If a digraph G contains an edge-kernel of size k then $\tau(G, \textit{all paths}) \geq k$.*

Proof. Let $K = \{e_1, e_2, \ldots, e_k\}$ be such an edge-kernel of size k. Assume for contradiction that there is a path-preserving labeling using on every edge at most $k-1$ labels. Then there is a path-preserving labeling that uses precisely $k-1$ labels on every edge (just extend the previous labeling by arbitrary labels). On every edge e_i, $1 \leq i \leq k$, sort the labels in an ascending order and denote by $\lambda_l(e)$ the lth smallest label of edge e; e.g. if an edge e has labels $\{1, 3, 7\}$, then $\lambda_1(e) = 1$, $\lambda_2(e) = 3$, and $\lambda_3(e) = 7$. Note that, by definition of an edge-kernel, all possible permutations of the edges in K appear in paths of G that should be preserved. We construct a permutation $\pi = (e_{j_1}, e_{j_2}, \ldots, e_{j_k})$ of the edges in K which cannot be time-respecting without using a kth label on some edge. As e_{j_1} use the edge with the maximum λ_1, that is $\arg\max_{e \in K} \lambda_1(e)$. Then as e_{j_2} use the edge with the maximum λ_2 between the remaining edges, that is $\arg\max_{e \in K \setminus \{e_{j_1}\}} \lambda_2(e)$, and define $e_{j_3}, e_{j_4}, \ldots$ analogously. It is not hard to see that π satisfies $\lambda_i(e_{j_i}) \geq \lambda_i(e_{j_{i+1}})$ for all $1 \leq i \leq k-1$. This, in turn, implies that for π to be time-respecting it cannot use the labels $\lambda_1, \ldots, \lambda_{i-1}$ at edge e_{j_i}, for all $i \geq 2$, which shows that at edge e_{j_k} it can use none of the $k-1$ available labels, thus a kth label is necessarily needed and the theorem follows. ■

Lemma 5 *If G is a complete digraph of order n then it has an edge-kernel of size $\lfloor n/2 \rfloor$.*

Proof. Note that $\lfloor n/2 \rfloor$ is the size of a maximum matching M of G. As all possible edges that connect the endpoints of the edges in M are available, M is an edge-kernel of size $\lfloor n/2 \rfloor$. ■

Now, Theorem 6 implies that a complete digraph of order n requires at least $\lfloor n/2 \rfloor$ labels on some edge in order for all paths to be preserved, that is $\lfloor n/2 \rfloor \leq \tau(G, \textit{all paths})$. At the same time we have the trivial upper bound $\tau(G, \textit{all paths}) \leq n-1$ which follows from the fact that the longest path of a clique is hamiltonian, thus has $n-1$ edges, and for any graph G the length of its longest path is an upper bound on $\tau(G, \textit{all paths})$.

The above, clearly remain true for the following (close to complete) bipartite digraph. There are two partitions $A = \{u_i : 1 \leq i \leq k\}$ and $B = \{v_i : 1 \leq i \leq k\}$ both of size k. The edge set consists of (u_i, v_i) for all i and (v_i, u_j) for all i, j. In words, from A to B we have only horizontal connections while from B to A we have all possible connections.

Lemma 6 *There exist planar graphs G with n vertices having edge-kernels of size $\Omega(n^{\frac{1}{3}})$.*

Proof. The proof is done by construction. Consider the grid graph $G = G_{2n^2,2n}$, i.e. G is formed as a part of the infinite grid having width of $2n^2$ vertices and height of $2n$ vertices. Note that G is a planar graph. For simplicity of the presentation, we consider the grid graph G on the Euclidean plane, where the vertices have integer coordinates and the lower left vertex has coordinates $(1, 1)$. Furthermore denote by $v_{i,j}$ the vertex of G that is placed on the point (i, j), where $1 \leq i \leq 2n^2$ and $1 \leq j \leq 2n$. For every $i \in \{1, 2, \ldots, n\}$ denote $p_i = v_{(2i-1)n,n}$ and $q_i = v_{(2i-1)n+1,n}$. We define the edge subset $S = \{e_i = p_iq_i : 1 \leq i \leq n\}$.

We now prove that S is an edge-kernel of G. Let $\pi = (e_{i_1}, e_{i_2}, \ldots, e_{i_n})$ be an arbitrary permutation of the edges of $S = \{e_1, e_2, \ldots, e_n\}$. We construct a simple path P in G that visits all the edges of S in the order of the permutation π. That is, we construct a path $P = (p_{i_1}, q_{i_1}, P_1, p_{i_2}, q_{i_2}, P_2, \ldots p_{i_{n-1}}, q_{i_{n-1}}, P_{n-1}, p_{i_n}, q_{i_n})$. In order to do so, it suffices to define iteratively the simple paths $P_1, P_2, \ldots, P_{n-1}$ such that no two of these paths share a common vertex. The path P_1 starts at q_{i_1} and continues upwards on the column of q_{i_1} in the grid, until it reaches the top $2n$th row of the grid. Then, if $i_2 > i_1$ (resp. if $i_2 < i_1$), the path P_1 continues on this top row to the right (resp. to the left), until it reaches the column of vertex p_{i_2} of the grid. Finally it continues downwards on this column until it reaches p_{i_2}, where P_1 ends.

Consider now an index $t \in \{2, 3, \ldots, n-1\}$. In a similar manner as P_1, the path P_t starts at vertex q_{i_t}. Then it continues upwards on the column of q_{i_t} in the grid as much as possible, such that it does not reach any vertex of a path P_k, where $k \leq t-1$. Note that, if no path P_k, $k \leq t-1$, passes through any vertex of the column of q_{i_t} in the grid, then the path P_t reaches the top $2n$th row of the grid in this column. On the other hand, note that, since $q_{i_t} = v_{(2i_t-1)n+1,n}$ and $t \leq n-1$, at most the upper $t-1 \leq n-2$ vertices of the column of q_{i_t} in the grid can possibly belong to a path P_k, where $k \leq t-1$. Thus the path P_t can always continue upwards from q_{i_t} by at least one edge. Let a_t be the uppermost vertex of P_t on the column of q_{i_t} of the grid (cf. Figure 6 for $t = 5$ and $e_{i_5} = e_1$).

Assume that $i_{t+1} > i_t$, i.e. vertex $p_{i_{t+1}}$ lies to the right of vertex q_{i_t} on the nth row of the grid. Then, the path P_t continues from vertex a_t to the right, as follows. If P_t can reach the column of p_{i_t} without passing through a vertex of a path P_k, $k \leq t-1$, then it does so; in this case the path P_t continues downwards until it reaches vertex p_{i_t}, where it ends (cf. Figure 6 for $t = 3$ and $e_{i_3} = e_3$). Suppose now that P_t can not reach the column of P_t without passing through a vertex of a path P_k, $k \leq t-1$ (cf. Figure 6 for $t = 5$ and $e_{i_5} = e_1$). Then, P_t continues on the row of vertex a_t to the right as much as possible (say, until vertex b_t), such that it does not reach any vertex of a path P_k, $k \leq t-1$. In this case the path P_t continues from vertex b_t downwards as much as possible until it reaches a vertex c_t that is not neighbored to its right to any vertex of a path P_k, $k \leq t-1$ (cf. Figure 6 for $t = 5$ and $e_{i_5} = e_1$). Furthermore P_t continues from vertex c_t to the right as much as possible until it reaches a vertex d_t that is not neighbored from above to any vertex of a path P_k, $k \leq t-1$. Then, P_t continues from d_t in a similar way until it reaches the column of vertex $p_{i_{t+1}}$ (cf. Figure 6 for $t = 5$, $e_{i_5} = e_1$, and $e_{i_6} = e_6$), and then it continues downwards until it reaches $p_{i_{t+1}}$, where P_t ends. Note that, by definition of the edge set S, there exist at least $2n$ columns of the grid between any two edges of the set S. Furthermore there exist $n-1$ rows of the grid below every edge of S. Thus, since there exist at most $t-1 \leq n-2$ previous paths P_k, $k \leq t-1$, it follows that there exists always enough space for the path P_t in the grid to (a) reach vertex d_t and (b) continue from d_t until it reaches vertex $p_{i_{t+1}}$, where P_t ends.

Assume now that $i_{t+1} < i_t$, i.e. vertex $p_{i_{t+1}}$ lies to the left of vertex q_{i_t} on the nth row of the grid. In this case, when we start the path P_t at vertex q_{i_t}, we first move one edge downwards and then two edges to the left (cf. Figure 6 for $t = 2$ and $e_{i_2} = e_5$, as well as for $t = 4$ and $e_{i_4} = e_4$). After that point we continue constructing the path P_t similarly to the case where $i_{t+1} > i_t$ (cf. Figure 6).

Therefore, we can construct in this way all the paths $P_1, P_2, \ldots, P_{n-1}$, such that no two of these paths share a common vertex, and thus the path $P = (p_{i_1}, q_{i_1}, P_1, p_{i_2}, q_{i_2}, P_2, \ldots p_{i_{n-1}}, q_{i_{n-1}}, P_{n-1}, p_{i_n}, q_{i_n})$ is a simple path of G that visits all the edges of S in the order of the permutation π. An example of the construction of such a path P is given in Figure 6.

In this example $S = (e_1, e_2, \ldots, e_6)$ and $\pi = (e_2, e_5, e_3, e_4, e_1, e_6)$. That is, using the above notation, $i_1 = 2$, $i_2 = 5$, $i_3 = 3$, $i_4 = 4$, $i_5 = 1$, and $i_6 = 6$. In this figure we also depict for $t = 5$ the vertices a_t, b_t, c_t that we defined in the above construction of the path P_{i_t}.

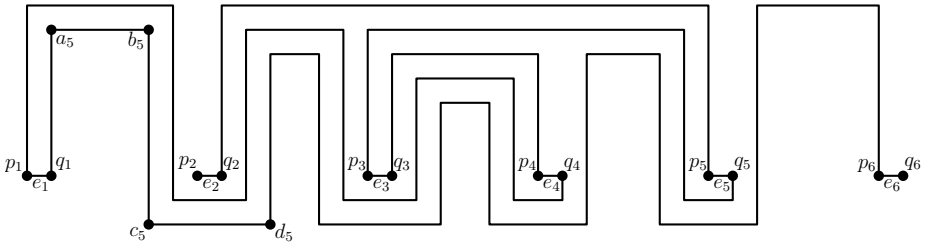

Figure 6: The edge-kernel $S = (e_1, e_2, \ldots, e_n\}$ of the grid graph with dimension $2n^2 \times 2n$, where $n = 6$, and a path P that visits the edges of S in the order of the permutation $\pi = (e_{i_1}, e_{i_2}, e_{i_3}, e_{i_4}, e_{i_5}, e_{i_6}) = (e_2, e_5, e_3, e_4, e_1, e_6)$.

Since such a path P exists for every permutation π of the edges of the set S, it follows by Definition 4 that S is an edge-kernel of G, where G is a planar graph. Finally, since $G = (V, E)$ has by construction $|V| = 4n^3$ vertices and $|S| = n$, it follows that the size of the edge-kernel S is $\Omega(|V|^{\frac{1}{3}})$. This completes the proof of the lemma. ■

5.3 Computing the Cost

5.3.1 Hardness of Approximation

Consider a boolean formula ϕ in conjunctive normal form with two literals in every clause (2-CNF). Let τ be a truth assignment of the variables of ϕ and $\alpha = (\ell_1 \vee \ell_2)$ be a clause of ϕ. Then α is *XOR-satisfied* (or *NAE-satisfied*) in τ, if one of the literals $\{\ell_1, \ell_2\}$ of the clause α is true in τ and the other one is false in τ. The number of clauses of ϕ that are XOR-satisfied in τ is denoted by $|\tau(\phi)|$. The formula ϕ is *XOR-satisfiable* (or *NAE-satisfiable*) if there exists a truth assignment τ of ϕ such that every clause of ϕ is XOR-satisfied in τ. The *Max-XOR* problem (also known as the *Max-NAE-2-SAT* problem) is the following maximization problem: given a 2-CNF formula ϕ, compute the greatest number of clauses of ϕ that can be simultaneously XOR-satisfied in a truth assignment τ, i.e. compute the greatest value for $|\tau(\phi)|$. The *Max-XOR(k)* problem is the special case of the Max-XOR problem, where every variable of the input formula ϕ appears in at most k clauses of ϕ. It is known that a special case of Max-XOR(3), namely the *monotone Max-XOR(3)* problem, is APX-hard (i.e. it does not admit a PTAS unless P=NP [9, 16]), as the next lemma states [1]. In this special case of the problem, the input formula ϕ is monotone, i.e. every variable appears not negated in the formula. The monotone Max-XOR(3) problem essentially encodes the *Max-Cut* problem on 3-regular (i.e. cubic) graphs, which is known to be APX-hard [1].

Lemma 7 ([1]) *The (monotone) Max-XOR(3) problem is APX-hard.*

Now we provide a reduction from the Max-XOR(3) problem to the problem of computing $\kappa(G, reach, d(G))$. Let ϕ be an instance formula of Max-XOR(3) with n variables $x_1, x_2, \ldots, x_n$ and m clauses. Since every variable x_i appears in ϕ (either as x_i or as $\overline{x_i}$) in at most 3 clauses, it follows that $m \leq \frac{3}{2}n$. We will construct from ϕ a graph G_ϕ having length of a directed cycle at most 2. Then, as we prove in Theorem 7, $\kappa(G_\phi, reach, d(G_\phi)) \leq 39n - 4m - 2k$ if and only if there exists a truth assignment τ of ϕ with $|\tau(\phi)| \geq k$, i.e. τ XOR-satisfies at least k clauses of ϕ. Since ϕ is an instance of Max-XOR(3), we can replace every clause $(\overline{x_i} \vee \overline{x_j})$ by the clause $(x_i \vee x_j)$ in ϕ, since $(\overline{x_i} \vee \overline{x_j}) = (x_i \vee x_j)$ in XOR. Furthermore, whenever $(\overline{x_i} \vee x_j)$ is a clause of ϕ, where $i < j$, we can replace this clause by $(x_i \vee \overline{x_j})$, since $(\overline{x_i} \vee x_j) = (x_i \vee \overline{x_j})$ in XOR. Thus, we can assume without loss of generality that every clause of ϕ is either of the form $(x_i \vee x_j)$ or $(x_i \vee \overline{x_j})$, where $i < j$.

For every $i = 1, 2, \ldots, n$ we construct the graph $G_{\phi,i}$ of Figure 7. Note that the diameter of $G_{\phi,i}$ is $d(G_{\phi,i}) = 9$ and the maximum length of a directed cycle in $G_{\phi,i}$ is 2. In this figure, we call the induced subgraph of $G_{\phi,i}$ on the 13 vertices $\{s^{x_i}, u_1^{x_i}, \ldots, u_6^{x_i}, v_1^{x_i}, \ldots, v_6^{x_i}\}$ the *trunk* of $G_{\phi,i}$. Furthermore, for every $p \in \{1, 2, 3\}$, we call the induced subgraph of $G_{\phi,i}$ on the 5 vertices $\{u_{7,p}^{x_i}, u_{8,p}^{x_i}, v_{7,p}^{x_i}, v_{8,p}^{x_i}, t_p^{x_i}, \}$ the *pth branch* of $G_{\phi,i}$. Finally, we call the edges $u_6^{x_i} u_{7,p}^{x_i}$ and $v_6^{x_i} v_{7,p}^{x_i}$ the *transition edges* of the pth branch of $G_{\phi,i}$. Furthermore, for every $i = 1, 2, \ldots, n$, let $r_i \leq 3$ be the number of clauses in which variable x_i appears in ϕ. For every $1 \leq p \leq r_i$, we assign the pth appearance of the variable x_i (either as x_i or as $\overline{x_i}$) in a clause of ϕ to the pth branch of $G_{\phi,i}$.

Consider now a clause $\alpha = (\ell_i \vee \ell_j)$ of ϕ, where $i < j$. Then, by our assumptions on ϕ, it follows that $\ell_i = x_i$ and $\ell_j \in \{x_j, \overline{x_j}\}$. Assume that the literal ℓ_i (resp. ℓ_j) of the clause α corresponds to the pth (resp. to the qth) appearance of the variable x_i (resp. x_j) in ϕ. Then we identify the vertices of the pth branch of $G_{\phi,i}$ with the vertices of the qth branch of $G_{\phi,j}$ as follows. If $\ell_j = x_j$ then we identify the vertices $u_{7,p}^{x_i}, u_{8,p}^{x_i}, v_{7,p}^{x_i}, v_{8,p}^{x_i}, t_p^{x_i}$ with the vertices $v_{7,q}^{x_j}, v_{8,q}^{x_j}, u_{7,q}^{x_j}, u_{8,q}^{x_j}, t_q^{x_j}$, respectively (cf. Figure 9(a)). Otherwise, if $\ell_j = \overline{x_j}$ then we identify the vertices $u_{7,p}^{x_i}, u_{8,p}^{x_i}, v_{7,p}^{x_i}, v_{8,p}^{x_i}, t_p^{x_i}$ with the vertices $u_{7,q}^{x_j}, u_{8,q}^{x_j}, v_{7,q}^{x_j}, v_{8,q}^{x_j}, t_q^{x_j}$, respectively (cf. Figure 9(b)). This completes the construction of the graph G_ϕ. Note that, similarly to the graphs $G_{\phi,i}$, $1 \leq i \leq n$, the diameter of G_ϕ is $d(G_\phi) = 9$ and the maximum length of a directed cycle in G_ϕ is 2. Furthermore, note that for each of the m clauses of ϕ, one branch of a gadget $G_{\phi,i}$ coincides with one branch of a gadget $G_{\phi,j}$, where $1 \leq i < j \leq n$, while every $G_{\phi,i}$ has three branches. Therefore G_ϕ has exactly $3n - 2m$ branches which belong to only one gadget $G_{\phi,i}$, and m branches that belong to two gadgets $G_{\phi,i}, G_{\phi,j}$.

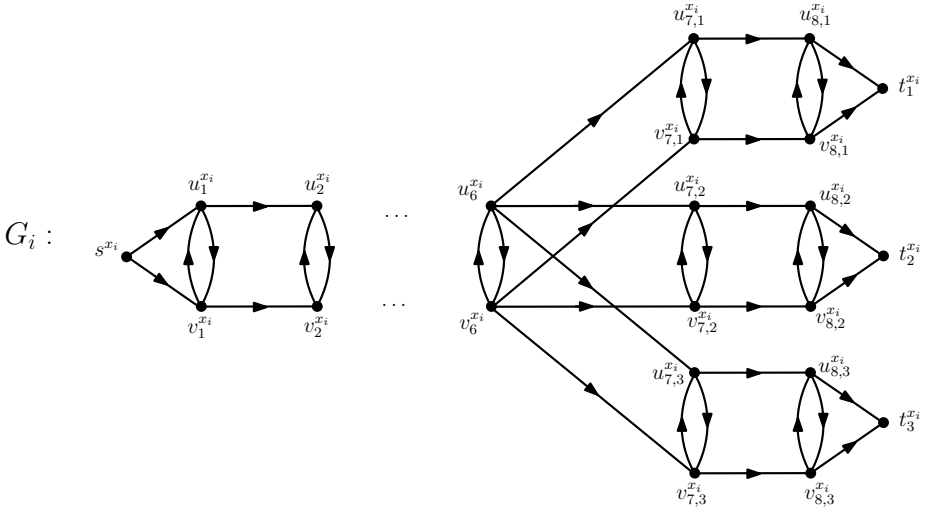

Figure 7: The gadget $G_{\phi,i}$ for the variable x_i.

Theorem 7 *There exists a truth assignment τ of ϕ with $|\tau(\phi)| \geq k$ if and only if $\kappa(G_\phi, reach, d(G_\phi)) \leq 39n - 4m - 2k$.*

Proof. ($\Rightarrow$) Assume that there is a truth assignment τ that XOR-satisfies k clauses of ϕ. We construct a labeling λ of G_ϕ with cost $39n - 4m - 2k$ as follows. Let $i = 1, 2, \ldots, n$. If $x_i = 0$ in τ, we assign labels to the edges of the trunk of $G_{\phi,i}$ as in Figure 8(a). Otherwise, if $x_i = 1$ in τ, we assign labels to the edges of the trunk of $G_{\phi,i}$ as in Figure 8(b). We now continue the labeling λ as follows. Consider an arbitrary clause $\alpha = (\ell_i \vee \ell_j)$ of ϕ, where $i < j$. Recall that $\ell_i = x_i$ and $\ell_j \in \{x_j, \overline{x_j}\}$. Assume that the literal ℓ_i (resp. ℓ_j) of the clause α corresponds to the pth (resp. to the qth) appearance of variable x_i (resp. x_j) in ϕ. Then, by the construction of G_ϕ, the pth branch of $G_{\phi,i}$ coincides with the qth branch of $G_{\phi,j}$.

Assume that $\ell_j = x_j$ (cf. Figure 9(a)). Then by our construction $u_{7,p}^{x_i} = v_{7,q}^{x_j}$, $u_{8,p}^{x_i} = v_{8,q}^{x_j}$, $v_{7,p}^{x_i} = u_{7,q}^{x_j}$, $v_{8,p}^{x_i} = u_{8,q}^{x_j}$, and $t_p^{x_i} = t_q^{x_j}$ (cf. Figure 9(a)). Let α be XOR-satisfied in τ, i.e. $x_i = \overline{x_j}$. If $x_i = \overline{x_j} = 0$ then we label the edges of the pth branch of $G_{\phi,i}$ (equivalently, the edges of the qth

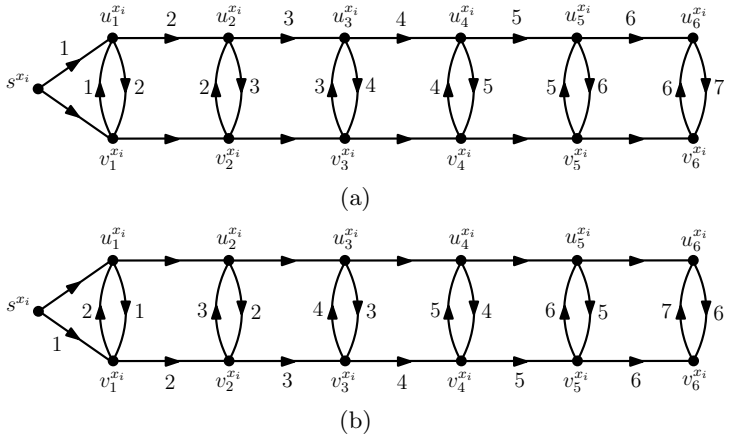

Figure 8: The labels of the edges of the trunk of $G_{\phi,i}$, where (a) $x = 0$ and (b) $x = 1$.

branch of $G_{\phi,j}$), the transition edges of the pth branch of $G_{\phi,i}$, and the transition edges of the qth branch of $G_{\phi,j}$, as illustrated in Figure 10(a). In the symmetric case where $x_i = \overline{x_j} = 1$ we label these edges in the same way as in Figure 10(a), with the only difference that we exchange the role of u's and v's. Let now α be XOR-unsatisfied in τ, i.e. $x_i = x_j$. If $x_i = x_j = 0$ then we label the edges of the pth branch of $G_{\phi,i}$ (equivalently, the edges of the qth branch of $G_{\phi,j}$), the transition edges of the pth branch of $G_{\phi,i}$, and the transition edges of the qth branch of $G_{\phi,j}$, as illustrated in Figure 10(b). In the symmetric case where $x_i = x_j = 1$ we label these edges in the same way as in Figure 10(b), with the only difference that we exchange the role of u's and v's. For the case where $\ell_j = \overline{x_j}$ we label the edges of Figure 9(b) similarly to the case where $\ell_j = x_j$ (cf. Figure 10).

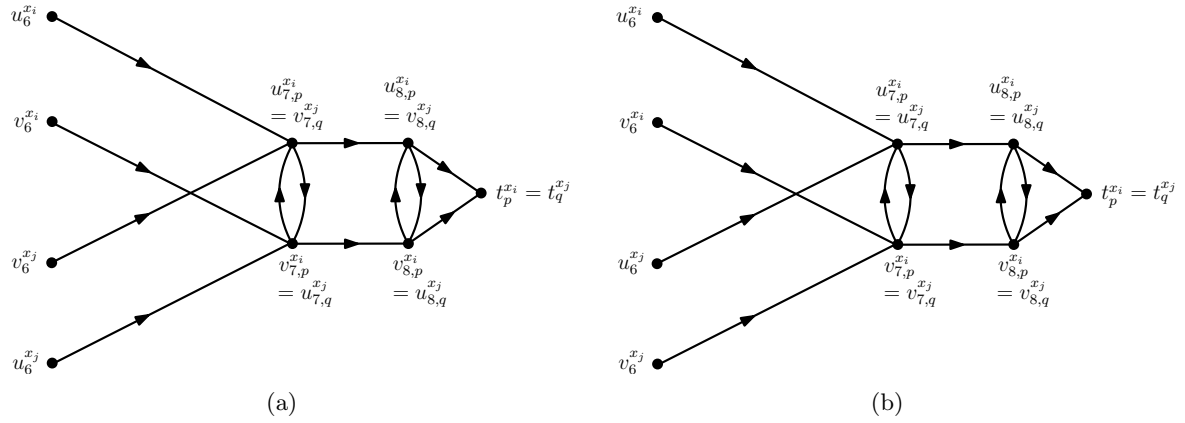

Figure 9: The gadgets for (a) the clause $(x_i \vee x_j)$ and (b) the clause $(x_i \vee \overline{x_j})$, where x_i appears in the pth branch of $G_{\phi,i}$ and x_j (resp. $\overline{x_j}$) appears in the qth branch of $G_{\phi,j}$.

Finally consider any of the $3n - 2m$ branches that belong to only one gadget $G_{\phi,i}$, where $1 \leq i \leq n$. Let this be the pth branch of $G_{\phi,i}$. If $x_i = 0$ then we label the edges of this branch and its transition edges as illustrated in Figure 10(a) (by ignoring in this figure the vertices $u_6^{x_j}, v_6^{x_j}$). In the symmetric case where $x_i = 1$, we label these edges in the same way, with the only difference that we exchange the role of u's and v's. This finalizes the labeling λ of G_ϕ. It is easy to check that λ preserves all reachabilities of G_ϕ and its greatest label is d.

Summarizing, for every $i \in \{1, 2, \ldots, n\}$, the edges of the trunk of $G_{\phi,i}$ are labeled with 18 labels (cf. Figure 8), and thus λ uses in total $18n$ labels for the trunks of all $G_{\phi,i}$, $i \in \{1, 2, \ldots, n\}$. Furthermore, for every $i \in \{1, 2, \ldots, n\}$ and every $p \in \{1, 2, 3\}$, λ uses 1 label for the two transition edges of the pth branch of $G_{\phi,i}$ (cf. Figure 10), and thus λ uses in total $3n$ labels for the transition edges of all $G_{\phi,i}$, $i \in \{1, 2, \ldots, n\}$. Moreover, for each of the $3n - 2m$ branches that belong to only one gadget $G_{\phi,i}$, where $1 \leq i \leq n$, λ uses 6 labels for the edges of this branch of $G_{\phi,i}$, and

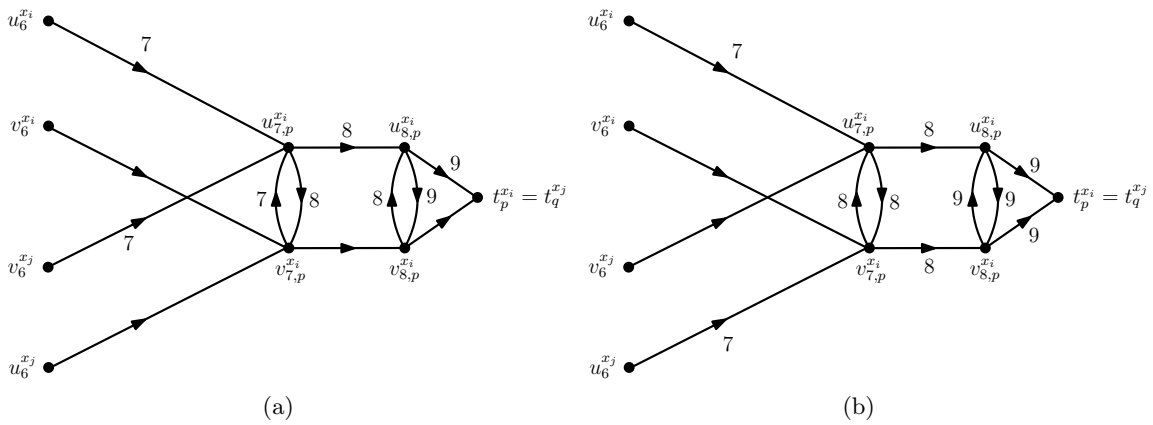

Figure 10: The labeling of the edges of Figure 9(a) for the clause $\alpha = (x_i \vee x_j)$, where (a) α is XOR-satisfied and $x_i = \overline{x_j} = 0$ in τ and (b) α is XOR-unsatisfied and $x_i = x_j = 0$ in τ.

thus λ uses in total $6(3n - 2m)$ labels for all these $3n - 2m$ branches. Finally consider any of the remaining m branches of G_ϕ, each of which corresponds to a clause α of ϕ (i.e. this branch belongs simultaneously to a gadget $G_{\phi,i}$ and a gadget $G_{\phi,j}$, where $1 \leq i < j \leq n$). If α is XOR-satisfied in τ, then λ uses 6 labels for the edges of this branch (cf. for example Figure 10(a)). Otherwise, if α is XOR-unsatisfied in τ, then λ uses 8 labels for the edges of this branch (cf. for example Figure 10(b)). Therefore, since τ XOR-satisfies by assumption k of the m clauses of ϕ, it follows λ uses in total $18n + 3n + 6(3n - 2m) + 6k + 8(m - k) = 39n - 4m - 2k$ labels, and thus $\kappa(G_\phi, reach, d(G_\phi)) \leq 39n - 4m - 2k$.

($\Leftarrow$) Assume that $\kappa(G_\phi, reach, d(G_\phi)) \leq 39n - 4m - 2k$ and let λ be a labeling of G_ϕ that maintains all reachabilities and has minimum cost (i.e. has the smallest number of labels); that is, $|\lambda| \leq 39n - 4m - 2k$. Let $i \in \{1, 2, \ldots, n\}$. Note that for every $z \in \{1, 2, \ldots, 6\}$, the vertices $u_z^{x_i}$ and $v_z^{x_i}$ reach each other in G_ϕ with a unique path (of length one). Therefore, each of the directed edges $\langle u_z^{x_i} v_z^{x_i} \rangle$ and $\langle v_z^{x_i} u_z^{x_i} \rangle$, where $z \in \{1, 2, \ldots, 6\}$, receives at least one label in every labeling, and thus also in λ. Similarly it follows that each of the directed edges $\langle u_{z,p}^{x_i} v_{z,p}^{x_i} \rangle$ and $\langle u_{z,p}^{x_i} v_{z,p}^{x_i} \rangle$, where $z \in \{7, 8\}$ and $p \in \{1, 2, 3\}$, receives at least one label in every labeling, and thus also in λ.

For every $i \in \{1, 2, \ldots, n\}$, define now the two paths $P_i = (s^{x_i}, u_1^{x_i}, u_2^{x_i}, \ldots, u_6^{x_i})$ and $Q_i = (s^{x_i}, v_1^{x_i}, v_2^{x_i}, \ldots, v_6^{x_i})$. Furthermore, for every $p \in \{1, 2, 3\}$, define the paths $P(i, p) = (P_i, u_{7,p}^{x_i}, u_{8,p}^{x_i}, t_p^{x_i})$ and $Q(i, p) = (Q_i, v_{7,p}^{x_i}, v_{8,p}^{x_i}, t_p^{x_i})$. Note that $P(i, p)$ and $Q(i, p)$ are the only two paths in G_ϕ from s^{x_i} to $t_p^{x_i}$ with distance $d(G_\phi) = 9$. Thus, since λ preserves all reachabilities of G_ϕ with maximum label 9, it follows that for every $i \in \{1, 2, \ldots, n\}$ and every $p \in \{1, 2, 3\}$, the edges of $P(i, p)$ or the edges of $Q(i, p)$ are labeled with the labels $1, 2, \ldots, 9$ in λ.

Assume that there exists an $i \in \{1, 2, \ldots, n\}$ such that all edges of the path P_i and all edges of the path Q_i are labeled in λ. Note that, if there exists no value $p \in \{1, 2, 3\}$ such that all edges of $P(i, p)$ (resp. of $Q(i, p)$) are labeled, then we can remove all labels from $P(i, p)$ (resp. from $Q(i, p)$) and construct another labeling λ' that still maintains all reachabilities of G_ϕ but has fewer labels than λ, which is a contradiction to the minimality assumption of λ. Therefore, there must exist values $p, q \in \{1, 2, 3\}$ such that all edges of $P(i, p)$ and all edges of $Q(i, q)$ are labeled in λ. Then, in both cases where $p = q$ and $p \neq q$, we modify λ into a labeling λ' as follows. We remove the labels from the seven edges of the path $(Q_i, v_{7,q}^{x_i})$, and we add labels (if they do not already have labels) to the six edges $\langle u_6^{x_i} u_{7,z}^{x_i} \rangle, \langle u_{7,z}^{x_i} u_{8,z}^{x_i} \rangle, \langle u_{8,z}^{x_i} t_z^{x_i} \rangle$, where $z \in \{1, 2, 3\} \setminus \{p\}$. Note that, in this new labeling λ', we can always preserve all reachabilities of the vertices by choosing the appropriate labels for the edges $\langle u_1^{x_i} v_1^{x_i} \rangle, \langle v_1^{x_i} u_1^{x_i} \rangle, \langle u_2^{x_i} v_2^{x_i} \rangle, \langle v_2^{x_i} u_2^{x_i} \rangle, \ldots, \langle u_6^{x_i} v_6^{x_i} \rangle$, $\langle v_6^{x_i} u_6^{x_i} \rangle, \langle u_{7,z}^{x_i} v_{7,z}^{x_i} \rangle, \langle v_{7,z}^{x_i} u_{7,z}^{x_i} \rangle, \langle u_{8,z}^{x_i} v_{8,z}^{x_i} \rangle, \langle v_{8,z}^{x_i} u_{8,z}^{x_i} \rangle$, where $z \in \{1, 2, 3\}$, cf. for example the labelings of Figures 8 and 10. However, by construction, the new labeling λ' uses a smaller number of labels than the initial labeling λ, which is a contradiction. Therefore, we may assume without loss of generality that for every $i \in \{1, 2, \ldots, n\}$, it is not the case that all edges of both paths P_i and Q_i

are labeled in λ, i.e. either all edges of P_i or all edges of Q_i are labeled in λ.

We now construct a truth assignment τ for the formula ϕ as follows. For every $i \in \{1, 2, \ldots, n\}$, if all edges of the path P_i are labeled in λ, then we define $x_i = 0$ in τ. Otherwise, if all edges of the path Q_i are labeled in λ, then we define $x_i = 1$ in τ. We will prove that $|\tau(\phi)| \geq k$, i.e. that τ XOR-satisfies at least k clauses of the formula ϕ.

Let $i \in \{1, 2, \ldots, n\}$. Recall that each of the directed edges $\langle u_z^{x_i} v_z^{x_i} \rangle$ and $\langle v_z^{x_i} u_z^{x_i} \rangle$, where $z \in \{1, 2, \ldots, 6\}$, receives at least one label in λ. Therefore, since all six edges of P_i or all six edges of Q_i are labeled in λ, it follows that λ uses for the trunk of $G_{\phi,i}$ at least 18 labels. Thus, λ uses in total at least $18n$ labels for the trunks of all $G_{\phi,i}$, $i \in \{1, 2, \ldots, n\}$.

Let now $p \in \{1, 2, 3\}$. Then, since $P(i,p) = (P_i, u_{7,p}^{x_i}, u_{8,p}^{x_i}, t_p^{x_i})$ and $Q(i,p) = (Q_i, v_{7,p}^{x_i}, v_{8,p}^{x_i}, t_p^{x_i})$ are the only two paths in G_ϕ from s^{x_i} to $t_p^{x_i}$ with distance $d(G_\phi) = 9$, it follows that λ uses at least one label for the pair of the transition edges $\{\langle u_6^{x_i} u_{7,p}^{x_i} \rangle, \langle v_6^{x_i} v_{7,p}^{x_i} \rangle\}$ of the pth branch of $G_{\phi,i}$. Thus, λ uses in total at least $3n$ labels for the transition edges of all $G_{\phi,i}$, $i \in \{1, 2, \ldots, n\}$.

Consider an arbitrary branch of G_ϕ, e.g. the pth branch of $G_{\phi,i}$, where $i \in \{1, 2, \ldots, n\}$ and $p \in \{1, 2, 3\}$. Since $P(i,p)$ and $Q(i,p)$ are the only two paths in G_ϕ from s^{x_i} to $t_p^{x_i}$ with distance $d(G_\phi) = 9$, it follows that λ assigns at least one label to each of the edges $\{\langle u_{7,p}^{x_i} u_{8,p}^{x_i} \rangle, \langle u_{8,p}^{x_i} t_p^{x_i} \rangle\}$, or at least one label to each of the edges $\{\langle v_{7,p}^{x_i} v_{8,p}^{x_i} \rangle, \langle v_{8,p}^{x_i} t_p^{x_i} \rangle\}$. Furthermore recall that each of the edges $\langle u_{z,p}^{x_i} v_{z,p}^{x_i} \rangle$ and $\langle v_{z,p}^{x_i} u_{z,p}^{x_i} \rangle$, where $z \in \{7, 8\}$, receives at least one label in λ. Therefore, λ uses at least 6 labels for an arbitrary branch of G_ϕ.

Consider now one of the clauses $\alpha = (\ell_i \vee \ell_j)$ of ϕ that are not XOR-satisfied in τ that we defined above. Note that there exist exactly $m - |\tau(\phi)|$ such clauses in ϕ. Let $i < j$, and thus $\ell_i = x_i$ and $\ell_j \in \{x_j, \overline{x_j}\}$. Assume that the literal ℓ_i (resp. ℓ_j) of the clause α corresponds to the pth (resp. to the qth) appearance of variable x_i (resp. x_j) in ϕ. Then, by the construction of G_ϕ, the pth branch of $G_{\phi,i}$ coincides with the qth branch of $G_{\phi,j}$. Suppose first that $\ell_j = x_j$. Then $x_i = x_j$, since α is not XOR-satisfied in τ. By the construction of the truth assignment τ from the labeling λ, it follows that either all edges of $P(i,p)$ and all edges of $P(j,q)$ are labeled in λ (in the case where $x_i = x_j = 0$), or all edges of $Q(i,p)$ and all edges of $Q(j,q)$ are labeled in λ (in the case where $x_i = x_j = 1$). Since $\ell_j = x_j$, note by the construction of G_ϕ that the last two edges of $P(i,p)$ are different from the last two edges of $P(j,q)$, while the last two edges of $Q(i,p)$ are different from the last two edges of $Q(j,q)$. Therefore, since each of the edges $\langle u_{z,p}^{x_i} v_{z,p}^{x_i} \rangle$ and $\langle v_{z,p}^{x_i} u_{z,p}^{x_i} \rangle$, where $z \in \{7, 8\}$, receives at least one label in λ, it follows that λ uses for the pth branch of $G_{\phi,i}$ (equivalently for the qth branch of $G_{\phi,j}$) at least 8 labels, if $\ell_j = x_j$.

Suppose now that $\ell_j = \overline{x_j}$. Then $x_i = \overline{x_j}$, since α is not XOR-satisfied in τ. Similarly to the case where $x_i = x_j$, it follows that either all edges of $P(i,p)$ and all edges of $Q(j,q)$ are labeled in λ (in the case where $x_i = \overline{x_j} = 0$), or all edges of $Q(i,p)$ and all edges of $P(j,q)$ are labeled in λ (in the case where $x_i = \overline{x_j} = 1$). Since $\ell_j = \overline{x_j}$, note by the construction of G_ϕ that the last two edges of $P(i,p)$ are different from the last two edges of $Q(j,q)$, while the last two edges of $Q(i,p)$ are different from the last two edges of $P(j,q)$. Therefore, since each of the edges $\langle u_{z,p}^{x_i} v_{z,p}^{x_i} \rangle$ and $\langle v_{z,p}^{x_i} u_{z,p}^{x_i} \rangle$, where $z \in \{7, 8\}$, receives at least one label in λ, it follows that λ uses for the pth branch of $G_{\phi,i}$ (equivalently for the qth branch of $G_{\phi,j}$) at least 8 labels, if $\ell_j = \overline{x_j}$.

Summarizing, λ uses in total at least $18n$ labels for the edges of the trunks of all $G_{\phi,i}$, at least $3n$ labels for the transition edges of all $G_{\phi,i}$, at least 6 labels for an arbitrary branch of G_ϕ, and at least 8 labels for each of the branches of G_ϕ that corresponds to a clause α of ϕ that is not XOR-satisfied in τ. Therefore, since G_ϕ has in total $3n-m$ branches and ϕ has $m-|\tau(\phi)|$ XOR-unsatisfied clauses in τ, it follows that λ uses at least $18n+3n+6(3n-m-(m-|\tau(\phi)|))+8(m-|\tau(\phi)|) = 39n-4m-2|\tau(\phi)|$ labels. However $|\lambda| \leq 39n - 4m - 2k$ by assumption. Therefore $39n - 4m - 2|\tau(\phi)| \leq |\lambda| \leq 39n - 4m - 2k$, and thus τ XOR-satisfies $|\tau(\phi)| \geq k$ clauses in ϕ. This completes the proof of the theorem. ■

Using Theorem 7, we are now ready to prove the main theorem of this section.

Theorem 8 (Hardness of Approximating the Temporal Cost) *The problem of computing $\kappa(G, reach, d(G))$ is APX-hard, even when the maximum length of a directed cycle in G is 2.*

Proof. Denote now by $\mathrm{OPT}_{\text{Max-XOR(3)}}(\phi)$ the greatest number of clauses that can be simultaneously XOR-satisfied by a truth assignment of ϕ. Then Theorem 7 implies that

$$\kappa(G_\phi, reach, d(G_\phi)) \leq 39n - 4m - 2 \cdot \mathrm{OPT}_{\text{Max-XOR(3)}}(\phi) \tag{1}$$

Note that a random assignment XOR-satisfies each clause of ϕ with probability $\frac{1}{2}$, and thus we can easily compute (even deterministically) an assignment τ that XOR-satisfies $\frac{m}{2}$ clauses of ϕ. Therefore $\mathrm{OPT}_{\text{Max-XOR(3)}}(\phi) \geq \frac{m}{2}$, and thus, since every variable x_i appears in at least one clause of ϕ, it follows that

$$\frac{n}{2} \leq \ m \leq 2 \cdot \mathrm{OPT}_{\text{Max-XOR(3)}}(\phi) \tag{2}$$

Assume that there is a PTAS for computing $\kappa(G, reach, d(G))$. Then, for every $\varepsilon > 0$ we can compute in polynomial time a labeling λ for the graph G_ϕ, such that

$$|\lambda| \leq (1+\varepsilon) \cdot \kappa(G_\phi, reach, d(G_\phi)) \tag{3}$$

Given such a labeling λ we can compute by the sufficiency part ($\Leftarrow$) of the proof of Theorem 7 a truth assignment τ of ϕ such that $39n - 4m - 2|\tau(\phi)| \leq |\lambda|$, i.e.

$$2|\tau(\phi)| \geq 39n - 4m - |\lambda| \tag{4}$$

Therefore it follows by (1), (2), (3), and (4) that

$$\begin{aligned}
2|\tau(\phi)| &\geq 39n - 4m - (1+\varepsilon) \cdot \kappa(G_\phi, reach, d(G_\phi)) \\
&\geq 39n - 4m - (1+\varepsilon) \cdot \big(39n - 4m - 2 \cdot \mathrm{OPT}_{\text{Max-XOR(3)}}(\phi)\big) \\
&= \varepsilon\,(4m - 39n) + 2(1+\varepsilon) \cdot \mathrm{OPT}_{\text{Max-XOR(3)}}(\phi) \\
&\geq \varepsilon\,(4m - 78m) + 2(1+\varepsilon) \cdot \mathrm{OPT}_{\text{Max-XOR(3)}}(\phi) \\
&\geq -74\varepsilon m + 2(1+\varepsilon) \cdot \mathrm{OPT}_{\text{Max-XOR(3)}}(\phi) \\
&\geq -74\varepsilon \cdot 2\mathrm{OPT}_{\text{Max-XOR(3)}}(\phi) + 2(1+\varepsilon) \cdot \mathrm{OPT}_{\text{Max-XOR(3)}}(\phi) \\
&= 2(1-73\varepsilon) \cdot \mathrm{OPT}_{\text{Max-XOR(3)}}(\phi)
\end{aligned}$$

and thus

$$|\tau(\phi)| \geq (1-73\varepsilon) \cdot \mathrm{OPT}_{\text{Max-XOR(3)}}(\phi) \tag{5}$$

That is, assuming a PTAS for computing $\kappa(G, reach, d(G))$, we obtain a PTAS for the Max-XOR(3) problem, which is a contradiction by Lemma 7. Therefore computing $\kappa(G, reach, d(G))$ is APX-hard. Finally, since the graph G_ϕ that we constructed from the formula ϕ has maximum length of a directed cycle at most 2, it follows that computing $\kappa(G, reach, d(G))$ is APX-hard even if the given graph G has maximum length of a directed cycle at most 2. ■

5.3.2 Approximating the Cost

In this section, we provide an approximation algorithm for computing $\kappa(G, reach, d(G))$, which complements the hardness result of Theorem 8. Given a digraph G define, for every $u \in V$, u's reachability number $r(u) = |\{v \in V : v \text{ is reachable from } u\}|$ and $r(G) = \sum_{u \in V} r(u)$, that is $r(G)$ is the total number of reachabilities in G.

Theorem 9 *There is an $\frac{r(G)}{n-1}$-factor approximation algorithm for computing $\kappa(G, reach, d(G))$ on any weakly connected digraph G.*

Proof. First of all, note that $\mathrm{OPT} \geq n-1$, where OPT is the cost of the optimal solution. The reason is that if a labeling labels less than $n-1$ edges then the subgraph of G induced by the labeled edges is disconnected (not even weakly connected) thus clearly fails to preserve some reachabilities. To see this, take any two components C_1 and C_2. G either has an edge from C_1 to C_2 or from C_2 to C_1 (or both). The two cases are symmetric so just consider the first one. Clearly some node from C_1 can reach some node from C_2 but this reachability has not been preserved by the labeling.

Now consider the following labeling algorithm.

1. For all $u \in V$, compute a BFS out-tree T_u rooted at u.
2. For all T_u, give to each edge at distance i from the root label i.
3. Output this labeling λ.

Clearly, the maximum label used by λ is $d(G)$: indeed if an edge e was assigned some label $l > d(G)$ then this would imply that on some BFS out-tree e appeared at distance $> d(G)$ which is a contradiction. Moreover, λ preserves all reachabilities as for every u the corresponding tree rooted at u reaches all nodes that are reachable from u and the described labeling clearly preserves the corresponding paths. Finally, we have that the cost paid by our algorithm is $\text{ALG} = |\lambda| = r(G)$. To see this, notice that for all u we use (i.e. we label) precisely $r(u)$ edges in T_u, thus, in total, we use $\sum_{u \in V} r(u) = r(G)$ edges by definition of $r(G)$.

We conclude that

$$\frac{\text{ALG}}{\text{OPT}} \leq \frac{r(G)}{n-1} \Rightarrow \text{ALG} \leq \frac{r(G)}{n-1}\text{OPT}$$

■

6 Conclusions and Further Research

There are many open problems related to the findings of the present work. We have considered several graph families in which the temporality of preserving all paths is very small (e.g. 2 for rings) and others in which it is very close to the worst possible (i.e. $\Omega(n)$ for cliques and $\Omega(n^{1/3})$ for planar graphs). There are still many interesting graph families to be investigated like regular or bounded-degree graphs. Moreover, though it turned out to be a generic lower-bounding technique related to the existence of a large edge-kernel in the underlying graph G, we still do not know whether there are other structural properties of the underlying graph that could cause a growth of the temporality (i.e. the absence of a large edge-kernel does not necessarily imply small temporality). Similar things hold also for the *reach* property. There are also many other natural connectivity properties subject to which optimization is to be performed that we haven't yet considered, like preserving a shortest path from u to v whenever v is reachable from u in G, or even depart from paths and require the preservation of more complex subgraphs (for some appropriate definition of "preservation"). Another interesting direction which we didn't consider in this work is to set the optimization criterion to be the age of λ e.g. w.r.t. the *all paths* or the *reach* connectivity properties. In this case, computing $\alpha(G, \text{all paths})$ is NP-hard, which can be proved by reduction from HAMPATH. On the positive side, it is easy to come up with a 2-factor approximation algorithm for $\alpha(G, reach, 2)$, where we have restricted the maximum number of labels of an edge (i.e. the temporality) to be at most 2. Additionally, there seems to be great room for approximation algorithms (or even randomized algorithms) for all combinations of optimization parameters and connectivity constraints that we have defined so far, or even polynomial-time algorithms for specific graph families. Finally, it would be valuable to consider other models of temporal graphs and in particular models with succinct representations, that is models in which the labels of every edge are provided by some short function associated to that edge (in contrast to a complete description of all labels). Such examples are several probabilistic models and several periodic models which are worth considering.

Acknowledgements. We would like to thank the anonymous reviewers of this article and its preliminary versions. Their thorough reading and comments have helped us to improve our work substantially.

References

[1] Paola Alimonti and Viggo Kann. Hardness of approximating problems on cubic graphs. In *Proceedings of the Third Italian Conference on Algorithms and Complexity (CIAC)*, pages 288–298, 1997.

[2] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, pages 235–253, March 2006.

[3] Chen Avin, Michal Koucký, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Proceedings of the 35th international colloquium on Automata, Languages and Programming (ICALP), Part I*, pages 121–132. Springer-Verlag, 2008.

[4] Kenneth A. Berman. Vulnerability of scheduled networks and a generalization of Menger's theorem. *Networks*, 28(3):125–134, 1996.

[5] Béla Bollobás. *Modern Graph Theory*. Springer, corrected edition, 1998.

[6] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems (IJPEDS)*, 27(5):387–408, 2012.

[7] Andrea E.F. Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding time in edge-markovian dynamic graphs. In *Proceedings of the 27th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 213–222, 2008.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[9] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*. SIAM Monographs on Discrete Mathematics and Applications, 2001.

[10] C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2013.

[11] Lisa Fleischer and Éva Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23(3):71–80, 1998.

[12] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, July 1987.

[13] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. In *Proceedings of the 12th annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 210–219, Philadelphia, PA, USA, 2001.

[14] Michal Katz, Nir A Katz, Amos Korman, and David Peleg. Labeling schemes for flow and connectivity. *SIAM Journal on Computing*, 34(1):23–40, 2004.

[15] David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the 32nd annual ACM symposium on Theory of computing (STOC)*, pages 504–513, 2000.

[16] Sanjeev Khanna, Rajeev Motwani, Madhu Sudan, and Umesh Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1999.

[17] Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM symposium on Theory of computing (STOC)*, pages 513–522, New York, NY, USA, 2010. ACM.

[18] George B. Mertzios, Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Temporal network optimization subject to connectivity constraints. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP), Part II*, pages 657–668, 2013.

[19] Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016.

[20] Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Mediated population protocols. *Theoretical Computer Science*, 412(22):2434–2450, May 2011.

[21] Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. *New Models for Population Protocols*. Morgan & Claypool, 2011.

[22] Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. *Journal of Parallel and Distributed Computing*, 74(1):2016–2026, 2014.

[23] Michael Molloy and Bruce Reed. *Graph colouring and the probabilistic method*, volume 23. Springer, 2002.

[24] Matthias Müller-Hannemann, Frank Schulz, Dorothea Wagner, and Christos D Zaroliagis. Timetable information: Models and algorithms. *ATMOS*, 4359:67–90, 2007.

[25] Regina O'Dell and Roger Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of the 2005 joint workshop on Foundations of mobile computing (DIALM-POMC)*, pages 104–110, 2005.

[26] Christian Scheideler. Models and techniques for communication in dynamic networks. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 27–49, 2002.

[27] Frank Schulz, Dorothea Wagner, and Karsten Weihe. Dijkstra's algorithm on-line: an empirical case study from public railroad transport. *Journal of Experimental Algorithmics (JEA)*, 5:12, 2000.

[28] B.B. Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.